ADITYA COLLEGE OF ENGINEERING
(An Autonomous Institution)
www.acem.ac.in

DEPARTMENT OF
C.S.E. - ARTIFICIAL INTELLIGENCE

Course Structure
&
Detailed Syllabus

For the students admitted to

B. Tech. Regular Four Year Degree Programme from the Academic Year 2026-27

and

B. Tech. Lateral Entry Scheme from the Academic Year 2027-28

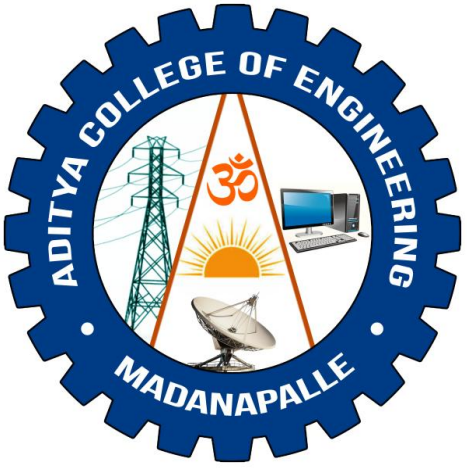
ADITYA COLLEGE OF ENGINEERING, MADANAPALLE

(UGC - Autonomous Institution & Accredited with NAAC A+ Grade)

Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu

Madanapalle - 517325, Annamayya (Dist.), A. P.

www.acem.ac.in



ACEM- R26 REGULATION

C.S.E. - ARTIFICIAL INTELLIGENCE

COURSE STRUCTURE AND SYLLABUS

for

B. Tech. (Regular – Full Time)

(Effective for the Students Admitted into I Year from the Academic Year 2026 - 27 onwards)

and

B. Tech. (Lateral Entry Scheme)

(Effective for the Students getting Admitted into II Year through Later Entry Scheme from the Academic Year 2027 - 28 onwards)

ADITYA COLLEGE OF ENGINEERING :: MADANAPALLE
Punganur Road, Valasapalle (Post), Madanapalle, Annamayya (Dist.,) – 517325.
(An Autonomous Institution)
DEPARTMENT OF CSE - ARTIFICIAL INTELLIGENCE

Institute Vision:

To impart quality in engineering education to meet the technological advances and industrial requirements with global standards.

Institute Mission:

- Provide quality technical education through skill-based trainings and promote research and development, and consultancy services.
- Offer state-of-the-art-infrastructure for supporting technological advances.
- Develop disciplined, creative and globally competent engineers.
- Equip and empower the faculty at all levels to promote innovations and technical advancements in various domains of engineering.

Department Vision:

To be a pioneering department that cultivates highly ethical, innovative, and globally competent AI professionals through superior teaching, cutting-edge research, and effective application of emerging technologies.

Department Mission:

- Develop AI professionals who are adept at emerging technologies and innovative in solving complex, real-world problems, ensuring they are well-prepared for a globalized workforce.
- Foster a culture of research and innovation among students and faculty, focusing on projects that address societal challenges and contribute to the advancement of AI.
- Embed ethical considerations and professional integrity into the AI curriculum and practice, guiding students to become responsible practitioners in the field.

Programme Outcomes (POs):

On Successful completion, the graduate will be able to,

- **PO1_Engineering Knowledge:** Apply knowledge of mathematics, natural sciences, engineering fundamentals, and an engineering specialization to a wide range of practical procedures and practices.
- **PO2_Problem Analysis:** Identify and analyse well-defined engineering problems and reach substantiated conclusions using codified methods of analysis specific to their field of activity.
- **PO3_Design/Development of Solutions:** Design solutions for well-defined technical problems and assist in the design of systems, components, or processes to meet specified needs, with appropriate consideration for public health and safety, as well as cultural, societal, and environmental factors.
- **PO4_Conduct Investigations of Complex Problems:** Conduct investigations of well-defined problems by locating and searching relevant codes and catalogues, and by conducting standard tests and measurements.

- **PO5_Engineering Tool Usage:** Apply appropriate techniques, resources, and modern computing, engineering and IT tools to well-defined engineering problems, while being aware of their limitations.
- **PO6_The Engineer and the World:** When solving well-defined engineering problems, evaluate sustainable development impacts to society, the economy, sustainability, health and safety, legal frameworks, and the environment.
- **PO7_Ethics:** Understand and commit to professional ethics and the norms of technician practice, including compliance with relevant laws. Demonstrate an understanding of the need for diversity and inclusion.
- **PO8_Individual and Collaborative Team Work:** Function effectively as an individual, and as a member or leader in diverse and inclusive teams, in multidisciplinary face-to-face/remote/distributed settings.
- **PO9_Communication:** Communicate effectively and inclusively on well-defined engineering activities with the engineering community and society at large by comprehending the work of others, documenting one's own work, and giving and receiving clear instructions.
- **PO10_Project Management and Finance:** Demonstrate awareness of engineering management principles as a member or leader of a technical team and apply them to manage projects in multidisciplinary environments.
- **PO11_Life-Long Learning:** Recognize the need for, and have the ability for, independent updating in the face of specialized technical knowledge.

Program Specific Outcomes (PSOs):

On Successful completion, the graduate CSE (Artificial Intelligence) will be able to,

- **PSO1:** Ability to understand and apply the concepts of Mathematics and Artificial Intelligence in analysing the real world problems and solving them.
- **PSO2:** Ability to design and develop computer programs / computer – based systems in the areas related to Artificial Intelligence, Machine Learning, Web / Mobile design and Data Analytics using software engineering principles and practices.

Programme Educational Objectives (PEOs):

After few years of graduation, the graduates of CSE (Artificial Intelligence) are expected to

- **PEO1:** Graduates of the program are adequately prepared to be employed in IT industries and Public Sector companies by forecasting a logical and practical approach to problem solving that would prepare them to function effectively as skilled computer engineers.
- **PEO2:** To impart students with solid foundation in mathematics, computing and core engineering fundamentals so as to help them to excel in their professional career or higher education.
- **PEO3:** To promote lifelong learning by encouraging research and an attitude to apply the basic theories learnt during their graduation, leading to the creativity and productivity in their respective fields.
- **PEO4:** To inculcate students with leadership qualities, communication skills and ethical behaviour as IT professionals that can lead to positive impact of technology on society.

B.Tech. – CSE (Artificial Intelligence)
Course Structure & Syllabus – ACEM R26 Regulations
(Applicable from the academic year 2026-27 onwards)

INDUCTION PROGRAMME

S. No.	Course Name	Category	L-T-P-C
1.	Physical Activities -- Sports, Yoga and Meditation, Plantation	MC	0-0-6-0
2.	Career Counselling	MC	2-0-2-0
3.	Orientation to all branches -- career options, tools, etc.	MC	3-0-0-0
4.	Orientation on admitted Branch -- corresponding labs, tools and platforms	EC	2-0-3-0
5.	Proficiency Modules & Productivity Tools	ES	2-1-2-0
6.	Assessment on basic aptitude and mathematical skills	MC	2-0-3-0
7.	Remedial Training in Foundation Courses	MC	2-1-2-0
8.	Human Values & Professional Ethics	MC	3-0-0-0
9.	Communication Skills -- focus on Listening, Speaking, Reading, Writing skills	BS	2-1-2-0
10.	Concepts of Programming	ES	2-0-2-0

Group - B
B.Tech. – I Year I Semester

S. No.	Category	Course Code	Name of the Course	L	T	Pr.	P	Credits
1.	BS	26BSHB101T	Linear Algebra & Calculus	3	0	2	0	4
2.	BS	26BSHB110T	Applied Chemistry for Computer Science	3	0	0	0	3
3.	HM	26HMHS101T	Communicative English	2	0	0	0	2
4.	ES	26ES05101T	Computational Thinking & Problem Solving with Python	3	0	0	0	3
5.	ES	26ES03103T	Design Thinking	1	0	4	0	3
6.	BS	26BSHB110P	Applied Chemistry for Computer Science Lab	0	0	0	3	1.5
7.	HM	26HMHS101P	Communicative English Lab	0	0	0	2	1
8.	ES	26ES05101P	Computational Thinking & Problem Solving with Python Lab	0	0	0	3	1.5
9.	HM	24HMHS102L	Health and Wellness, Yoga and Sports	0	0	0	1	0.5
10.	MC	26ES05102A	Cybersecurity Awareness for Engineers	2	0	0	0	0
			Total	**12+2**	**0**	**6**	**9**	**19.5**

B.Tech. – I Year II Semester

S. No.	Category	Course Code	Name of the Course	L	T	Pr.	P	Credits
1.	BS	26BSHB102T	Differential Equations & Vector Calculus	3	0	2	0	4
2.	BS	26BSHB106T	Physics for Advanced Computing	2	1	0	0	3
3.	ES	26ES03101T	Engineering Graphics	1	0	4	0	3
4.	ES	26ES31101T	AI Tools and Applications	1	0	4	0	3
5.	PC	26PC05101T	Data Structures & Efficient Problem Solving	3	0	0	0	3
6.	BS	26BSHB106P	Physics for Advanced Computing Lab	0	0	0	3	1.5
7.	PC	26PC05101P	Data Structures & Efficient Problem-Solving Lab	0	0	0	2	1
8.	ES	26ES03102P	Engineering Workshop	0	0	0	3	1.5
9.	HM	26HMHS103L	NSS / NCC / Scouts & Guides / Community Service	0	0	0	1	0.5
			Total	**10**	**1**	**10**	**9**	**20.5**

B.Tech. – II Year I Semester

S. No.	Category	Course Code	Name of the Course	L	T	Pr.	P	Credits
1.	BS	26BSHB213T	Biology for Engineers	2	0	0	0	2
2.	HM	26HMHS204T	Universal Human Values – Understanding Harmony & Ethical Human Conduct	3	0	0	0	3
3.	PC	26PC31201T	Fundamentals of Artificial Intelligence	2	0	2	0	3
4.	EC	26EC04203T	Digital Logic and Computer Organization	3	0	0	0	3
5.	PC	26PC05202T	Algorithms Design & Analysis	3	0	0	0	3
6.	PC	26PC05203T	Object-Oriented Design & Programming	3	0	0	0	3
7.	EC	26EC04203P	Digital Logic and Computer Organization Lab	0	0	0	2	1
8.	PC	26PC05202P	Algorithms Design & Analysis Lab	0	0	0	3	1.5
9.	PC	26PC05203P	Object-Oriented Design & Programming Lab	0	0	0	3	1.5
10.	SE	26SE31201S	AI Assisted Application Development	1	0	0	2	2
11.	MC	26HMHS206A	Technical Paper Writing and IPR	2	0	0	0	0
			Total	**17+2**	**0**	**2**	**10**	**23**

B. Tech. – II Year II Semester

S. No.	Category	Course Code	Name of the Course	L	T	Pr.	P	Credits
1.	BS	26BSHB212T	Probability and Statistics	2	0	2	0	3
2.	HM	26HMHS205T	Managerial Economics and Financial Analysis	2	0	0	0	2
3.	ES	26ES01101T	Sustainability and Green Engineering (AI Integrated)	2	0	0	0	2
4.	PC	26PC05205T	Intelligent Database Management Systems	3	0	0	0	3
5.	PC	26PC31202T	Intelligent Data Processing and System Integration	3	0	0	0	3
6.	PC	26PC31203T	Computer Architecture for Intelligent Systems	2	0	2	0	3
7.	PC	26PC31202P	Intelligent Data Processing and System Integration Lab	0	0	0	3	1.5
8.	PC	26PC05205P	Intelligent Database Management Systems Lab	0	0	0	3	1.5
9.	SE	26SE30201S	Data Analytics using Python	1	0	0	2	2
10.	MC	26BSHB214A	Environmental Science	2	0	0	0	0
			Total	**15+2**	**0**	**4**	**8**	**21**
	Mandatory Community Service Project (26IPIN301L) of 08 Weeks duration during Summer Vacation							

B. Tech. – I Year
I Semester

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26BSHB101T	BS	**Linear Algebra & Calculus** (Common to all branches of Engineering)	3	0	2	0	4

Pre-Requisites: Basic algebra, trigonometry, coordinate geometry, differential and integral calculus, vectors, matrices, determinants, and functions of several variables.

Course Objectives:

- To build strong foundations in linear algebra and multivariable calculus with direct engineering relevance.
- To integrate analytical methods with AI-assisted computational tools for visualization, simulation, and verification.
- To develop outcome-oriented problem-solving skills aligned with NEP 2020 and modern engineering applications.
- To apply matrix methods, eigenanalysis, and multivariable calculus to solve engineering problems.
- To use computational tools for approximation, optimization, differentiation, and integration.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply matrix algebra techniques to solve systems of linear equations and analyze matrix properties relevant to engineering models. (L3)

CO2: Analyze eigenvalue problems, diagonalization, and quadratic forms to interpret linear transformations and engineering systems. (L4)

CO3: Apply mean value theorems, Taylor and Maclaurin series, and curvature concepts to approximate and interpret engineering functions. (L3)

CO4: Analyze multivariable functions using partial derivatives, Jacobians, gradients, and optimization techniques. (L4)

CO5: Evaluate areas, volumes, and coordinate transformations using multiple integrals in engineering contexts. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	1		1				2	2	1
CO2	3	3	3	2	1		1				2	2	1
CO3	3	2	2	2	1		1				2	2	1
CO4	3	3	3	3	2		1				2	2	2
CO5	3	3	2	3	3		1				2	2	1

Unit I: Matrix Algebra and Linear Systems:

Core Content: Matrices, operations, transpose, inverse, types of matrices, symmetric and skew-symmetric matrices; orthogonal, Hermitian, skew-Hermitian and unitary matrices. determinant properties; rank of a matrix; echelon and normal forms; system of linear equations; consistency conditions; Gauss elimination and Gauss-Jordan methods; iterative methods: Jacobi and Gauss-Seidel;

AI Integration: Python-based solution of large linear systems using NumPy; visualization of convergence in iterative methods; AI-assisted debugging of matrix algorithms; using ChatGPT/Copilot to explain row operations and rank reduction.

Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib) on Google Colab; SymPy for symbolic matrix operations; Jupyter Notebook; Scilab.

Unit II: Eigenvalues, Eigenvectors, Diagonalization & Quadratic Forms:

Core Content: Eigenvalues and eigenvectors; Properties, characteristic equation; Cayley-Hamilton theorem; diagonalization; similarity transformation; powers and inverse of matrices using Cayley-Hamilton theorem; quadratic forms; canonical form; reduction by orthogonal transformation; rank, index, signature; Nature of Quadratic form;

AI Integration: AI-assisted eigenvalue computation and verification; Python/SciPy visualization of eigen modes and quadratic surfaces; use of AI tools to explain diagonalization and matrix classification; numerical checking of Cayley-Hamilton theorem.

Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib); SymPy; GNU Octave; Google Colab.

Unit III: Single Variable Calculus and Series Expansions:

Core Content: Rolle's theorem; Lagrange's mean value theorem; Cauchy's mean value theorem; Taylor's and Maclaurin's series with remainder; approximation of functions; error estimation; curvature of plane curves; radius and centre of curvature; circle of curvature; applications to engineering approximations and curve design.

AI Integration: SymPy-based Taylor and Maclaurin expansion generation; plotting truncation error and convergence; AI-supported derivation checking for mean value theorems; curvature plot visualization and interpretation.

Free/Open-Source AI Tools: Python (SymPy, NumPy, Matplotlib); Google Colab; Jupyter Notebook; WolframAlpha free tier for verification.

Unit IV: Multivariable Differentiation and Optimization:

Core Content: Functions of several variables; limits and continuity; partial derivatives; higher order partial derivatives; total derivative; chain rule; gradient; tangent plane and normal line; Jacobians; functional dependence; Taylor expansion of two-variable functions; maxima and minima; second derivative test; Lagrange multipliers and constrained optimization; divergence and curl.

AI Integration: Gradient field visualization using Python; AI-assisted Jacobian computation and verification; constrained optimization using SciPy; use of AI tools to interpret saddle points, extrema, and optimization constraints.

Free/Open-Source AI Tools: Python (NumPy, SciPy, SymPy, Matplotlib); Google Colab; Jupyter Notebook; SciPy optimize module.

Unit V: Multiple Integrals and Coordinate Transformations:

Core Content: Double integrals; evaluation over type I and type II regions; change of order of integration; triple integrals; change of variables; polar, cylindrical and spherical coordinates; area and volume computation; mass, centroid and moment calculations; engineering applications of multiple integration.

AI Integration: Numerical evaluation of double and triple integrals using SciPy; 2D and 3D region visualization; AI-assisted generation of integration code; interpreting Jacobian as geometric scaling factor in transformed coordinates.

Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib); Google Colab; SymPy; Jupyter Notebook; 3D plotting tools.

Text Books:

1. B. S. Grewal, Higher Engineering Mathematics, 44th Edition, Khanna Publishers, 2017.
2. Erwin Kreyszig, Advanced Engineering Mathematics, 10th Edition, Wiley, 2018.

3. G. B. Thomas, M. D. Weir, J. Hass, Thomas' Calculus, 14th Edition, Pearson, 2018.
4. R. K. Jain and S. R. K. Iyengar, Advanced Engineering Mathematics, 5th Edition, Alpha Science International, 2021.

Reference Books:

1. Gilbert Strang, Introduction to Linear Algebra, 5th Edition, Wellesley-Cambridge Press, 2016.
2. David C. Lay, Steven R. Lay, Judi J. McDonald, Linear Algebra and Its Applications, 5th Edition, Pearson, 2015.
3. George B. Arfken, Hans J. Weber, Frank E. Harris, Mathematical Methods for Physicists, 7th Edition, Academic Press, 2012.
4. Dennis G. Zill and Warren S. Wright, Advanced Engineering Mathematics, 6th Edition, Jones & Bartlett, 2018.
5. E. Kreyszig, Elementary Linear Algebra, 2nd Edition, Wiley, 1983.

Web Resources & NPTEL Links:

1. NPTEL: Linear Algebra – https://nptel.ac.in
2. MIT OpenCourseWare – Linear Algebra: https://ocw.mit.edu
3. MIT OpenCourseWare – Multivariable Calculus: https://ocw.mit.edu
4. Python SymPy Documentation: https://docs.sympy.org
5. Google Colab for Python: https://colab.research.google.com
6. SciPy Documentation: https://docs.scipy.org
7. NumPy Documentation: https://numpy.org/doc

Recommended Free & Open-Source AI/Computational Tools:

1. Python (NumPy, SciPy, Matplotlib) – FREE on Google Colab for matrix computation, visualization, and numerical analysis.
2. SymPy (free, open-source) – For symbolic algebra, calculus, matrix operations, and verification.
3. scikit-learn (free, open-source) – For regression, classification, and exploratory predictive modeling.
4. Jupyter Notebook (free, open-source) – For interactive computation and lab documentation.
5. OpenCV (free, open-source) – For image-based experimentation and visualization support where needed.
6. GNU Octave (free, open-source) – MATLAB-like numerical computing for matrix and calculus workflows.
7. TensorFlow / Keras (free, open-source) – For optional AI demonstrations in regression and classification tasks.

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26BSHB110T	BS	**Applied Chemistry for Computer Science** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	3	0	0	0	3

Pre-Requisites: Basic Chemistry (Higher Secondary level).

Course Objectives:

- To understand the fundamentals of quantum chemistry including wave-particle duality, quantum numbers, and molecular orbital theory relevant to computing materials.
- To comprehend the chemistry and electronic properties of semiconductor materials and the chemical processes involved in semiconductor device fabrication.
- To evaluate electrochemical principles governing batteries, supercapacitors, and fuel cells as energy storage technologies for computing systems.
- To analyze corrosion mechanisms in electronic devices and design appropriate protective techniques for PCBs, solder joints, and interconnects.
- To explore the chemistry, synthesis methods, and properties of nanomaterials used in nanoelectronics, quantum computing, and sustainable electronics.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply quantum chemistry principles to analyze molecular systems and explain molecular orbital theory for computing materials. (L3)

CO2: Analyze semiconductor materials, doping chemistry, and fabrication processes used in modern electronic devices. (L4)

CO3: Evaluate energy storage technologies including batteries, supercapacitors, and fuel cells for computing and data centre applications. (L5)

CO4: Analyze corrosion mechanisms in electronic devices and recommend appropriate protective techniques for PCBs and interconnects. (L4)

CO5: Design advanced materials solutions using nanotechnology principles and AI-assisted tools for nanoelectronics and quantum computing. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	3	2	1	1				2	2	1
CO2	3	3	3	2	3	1	1				2	2	1
CO3	3	3	3	2	3	2	1				2	2	1
CO4	2	3	2	3	3	2	1				2	2	1
CO5	3	2	3	3	3	2	1				3	2	2

Unit I: Quantum Chemistry:

Core Content: Fundamentals of quantum chemistry — black body radiation, photoelectric effect, wave-particle duality, de-Broglie wave equation, Heisenberg's uncertainty principle. Fundamentals of quantum mechanics, Schrödinger wave equation, significance of Ψ and Ψ^2. Quantum numbers. Particle in a one-dimensional box. Molecular orbital theory — bonding in homo and hetero nuclear diatomic molecules — energy level diagrams of O_2 and CO. π-molecular orbitals of butadiene and benzene, calculation of bond order.

AI Integration: Simulation of quantum systems using Python (NumPy, SciPy); visualization of molecular orbitals and energy levels using computational chemistry tools (Google Colab); introduction to quantum computing simulations using Qiskit (introductory level).

Free/Open-Source AI Tools: Python (NumPy, SciPy), Google Colab, Qiskit.

Unit II: Semiconductor Chemistry:

Core Content: Atomic structure and chemical bonding relevant to semiconductors, Band theory: valence band, conduction band, band gap energy. Intrinsic and extrinsic semiconductors: silicon, germanium, gallium arsenide. Doping chemistry: n-type (phosphorus, arsenic) and p-type (boron, aluminium) dopants. Chemical vapour deposition (CVD) for thin film formation; silicon dioxide (SiO_2) formation and properties.

Chemistry of compound semiconductors: GaAs, InP, SiC.

AI Integration: AI-based prediction of semiconductor properties using ML models; process parameter optimization in semiconductor fabrication using machine learning; Python-based band gap prediction using regression models.

Free/Open-Source AI Tools: Python, TensorFlow, Scilab.

Unit III: Electrochemistry and Energy Storage:

Core Content: Electrochemical cells: galvanic cells, electrode potential, Nernst equation and problems; potentiometry (redox reactions); conductometry (strong acid vs. strong base, weak acid vs. strong base). Primary batteries: Zinc-air battery, Sodium-air battery. Secondary batteries: chemistry of charging and discharging of lead-acid, nickel-cadmium, nickel-metal hydride. Lithium-ion batteries: construction, working, and applications of $LiCoO_2$. Supercapacitors: definition, classification, and applications. Fuel cells: construction, working, and applications of hydrogen fuel cells and direct methanol fuel cells (DMFC). Battery management systems: overcharge protection and thermal management.

AI Integration: AI models for battery health prediction and state-of-charge optimization using scikit-learn; data analysis for energy efficiency in data centres using Python (Pandas); ML-based prediction of electrochemical impedance spectra.

Free/Open-Source AI Tools: Python (Pandas, scikit-learn), Google Colab.

Unit IV: Chemistry of Corrosion and Protective Techniques:

Core Content: Corrosion: definition, examples, types — chemical corrosion and electrochemical corrosion; Pilling-Bedworth rule. Galvanic corrosion; concentration cell corrosion; factors affecting corrosion. Deal-Grove model: thermal oxidation of silicon; passivation layers — silicon dioxide, silicon nitride, and aluminium oxide. Corrosion of printed circuit boards (PCBs): copper oxidation, tin whiskers; corrosion in solder joints and interconnects. Protective techniques: proper design, using pure metals, metal alloys, inhibitors; cathodic protection — sacrificial anodic protection, impressed current cathodic protection. Electroplating (nickel) and electroless plating (copper).

AI Integration: Computer vision (OpenCV) for automated defect detection in PCBs; AI-based prediction of corrosion failure in electronic components using TensorFlow; image classification models for identifying corrosion types from microscope images.

Free/Open-Source AI Tools: OpenCV, TensorFlow, Python.

Unit V: Chemistry of Nanomaterials:

Core Content: Introduction to nanochemistry: size-dependent properties, quantum confinement. Carbon nanomaterials: fullerenes, carbon nanotubes (CNTs), graphene. Synthesis methods: chemical vapour deposition (CVD), sol-gel process. Quantum dots: synthesis, properties, applications in displays and quantum computing. Metallic nanoparticles: preparation and applications of gold, silver, and copper nanoparticles. Applications: nanoelectronics, interconnects, sensors (optical and biosensors), memory devices. Spintronics materials: magnetic nanoparticles, topological insulators. Chemistry of quantum computing materials: superconducting materials (NbTi, Nb_3Sn); ion trap materials.

AI Integration: AI-based nanomaterial property prediction using machine learning (Python, TensorFlow); simulation of nano-devices and quantum materials using open-source tools; generative AI for exploring nanomaterial design spaces.

Free/Open-Source AI Tools: Python, TensorFlow, open-source materials simulation platforms.

Text Books:

1. Jain P.C. and Jain M., Engineering Chemistry, 17th Edition, Dhanpat Rai Publishing Company, 2020.
2. Agarwal S., Engineering Chemistry, 5th Edition, Cambridge University Press, 2026.

Reference Books:

1. Linden, D., & Reddy, T. B. (Eds.). (2001). Handbook of Batteries (3rd ed.). McGraw-Hill.
2. Rao, C. N. R., Müller, A., & Cheetham, A. K. (Eds.). (2007). The Chemistry of Nanomaterials: Synthesis, Properties and Applications (Vols. 1–2). Wiley-VCH.
3. Poole, C. P., & Owens, F. J. (2003). Introduction to Nanotechnology. John Wiley & Sons.
4. Sharma, B. K., & Gaur, H. (2019). Engineering Chemistry. Krishna Prakashan Media.
5. Dara, S. S., & Umare, S. S. (2018). Engineering Chemistry (15th ed.). S. Chand Publishing.

Online Resources:

1. NPTEL Engineering Chemistry: https://nptel.ac.in/
2. IBM Qiskit Learning: https://learning.quantum.ibm.com/
3. Google Colab: https://colab.research.google.com/
4. OpenCV Python Documentation: https://docs.opencv.org/
5. Materials Project (open nanomaterials database): https://materialsproject.org/

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26HMHS101T	HM	**Communicative English** (Common for all branches of Engineering)	2	0	0	0	2

Pre-Requisites: Basic English (Intermediate level).

Course Objectives:

- To develop strong foundations in English grammar, vocabulary, phonetics, and communication models essential for professional contexts.
- To develop listening, speaking, reading, and writing skills using structured activities and AI-powered tools.
- To enable students to compose professional emails, reports, and presentations using AI writing assistants.
- To develop critical analysis of communication styles, barriers, and discourse structures using AI tools.
- To cultivate responsible AI usage for language learning — verifying AI outputs, maintaining academic integrity, and using AI as an aid not a substitute.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply the principles of communication, grammatical structures, tenses, subject–verb agreement, phonetic patterns, and vocabulary skills to identify and correct language errors and demonstrate effective spoken and written communication using AI-assisted tools. (L3)

CO2: Apply effective listening, speaking, group discussion, presentation, and debate techniques to organize and deliver clear, confident, and persuasive oral communication using AI-assisted tools for speech analysis, transcription, and visual presentation design. (L3)

CO3: Apply reading and writing strategies to comprehend, organize, and produce clear paragraphs, creative content, movie reviews, and blogs using AI-assisted paraphrasing, readability, simplification, and content-structuring tools. (L3)

CO4: Analyze professional and business communication requirements to evaluate and improve the structure, tone, formality, and effectiveness of emails, letters, reports, summaries, resumes, and LinkedIn profiles using AI-assisted communication tools. (L4)

CO5: Evaluate the effectiveness, accuracy, and ethical implications of AI-assisted language learning and communication tools, with regard to translation, plagiarism, academic integrity, bias, privacy, misinformation, and responsible digital communication. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1						1	1	1	3		2		
CO2						2	1	2	3		2		
CO3						2	2	3	3		2		
CO4						2	2	2	3		2		
CO5						2	3	2	3		3		

Unit I: Foundations of English Communication:

Core Content: Basics of Communication: process, types (verbal/non-verbal), and barriers. Parts of Speech: nouns, pronouns, verbs, adjectives, adverbs. Tenses: simple, continuous, perfect, and perfect continuous. Subject-verb agreement and correction of sentences.

AI Integration: Grammar explanation, error identification, and instant Q&A on language rules, Real-time grammar, spelling, and punctuation correction with detailed explanations, Phonetics practice – voice recording for pronunciation and intonation feedback, Word-of-the-day, etymology,

and interactive vocabulary quizzes.
Free/Open-Source AI Tools: Grammarly (free); ChatGPT / Gemini (free tiers); Google Speech-to-Text (free); Merriam-Webster Vocabulary Builder (free)

Unit II: Listening and Speaking Skills:
Core Content: Principles of Effective Speaking and Active Listening. Group Discussion (GD): techniques, etiquette, and evaluation criteria. Presentation Skills: design principles, slide preparation, and delivery techniques. Debate: argument construction, evidence use, and rebuttal techniques.
AI Integration: AI-powered presentation design – structured, visual, professional slides, Fluency analysis, filler-word detection, pacing, and confidence scoring, Auto-transcription and keyword analysis of spoken communication, Visual communication design – infographics, posters, and slide aesthetics
Free/Open-Source AI Tools: Gamma.app (free); Orai AI Speech Coach (free); Otter.ai (free tier); Canva AI (free)

Unit III: Reading Comprehension & Writing Skills:
Core Content: Reading Strategies: skimming, scanning, intensive and extensive reading. Comprehension passages, synonyms, antonyms, words often confused, homonyms, homophones, and homographs. Paragraph Writing: topic sentence, supporting details, clincher sentence. Creative Writing — features and techniques — movie reviews and blogs.
AI Integration: AI paraphrasing tool – rewrite passages at varying complexity levels, Readability scoring, passive voice detection, and sentence simplification, Essay outline generation, argument structuring, and content brainstorming, Vocabulary simplification for improving comprehension of complex texts
Free/Open-Source AI Tools: QuillBot (free); Hemingway Editor (free online); ChatGPT (free); Rewordify (free)

Unit IV: Professional & Business Communication:
Core Content: Email Writing and Letter Writing: formal and informal. Short Report Writing: structure, types (formal/informal), and format. Note-Making and Summarization Techniques. Resume / CV Writing and LinkedIn Profile Optimization.
AI Integration: Draft and refine professional emails, cover letters, reports, and proposals, AI-powered resume builder with ATS-optimized templates, Report structuring, agenda writing, and knowledge management, Tone detection, formality check, and professional register evaluation
Free/Open-Source AI Tools: ChatGPT (free); Kickresume (free tier); Grammarly (free); Notion AI (free tier)

Unit V: Use of AI in Language Learning:
Core Content: Artificial Intelligence in Communication: meaning, basics, benefits and limitations of AI in human interaction. AI chatbots and virtual assistants in education: grammar correction, vocabulary building, AI-based speaking and pronunciation tools, writing assistants, e-portfolio creation. AI in Global Communication: automated translation, summarizing large texts, finding information in different languages. Digital Communication and AI Ethics: responsible use of AI in education, plagiarism and academic honesty, fake information and AI-generated content, privacy, bias, and ethical communication online.
AI Integration: Plagiarism detection, paraphrase checker, and academic integrity analysis, High-accuracy translation supporting intercultural communication understanding, AI literacy modules –

understanding AI bias, ethics, and responsible usage, E-portfolio creation platform for showcasing communication artifacts

Free/Open-Source AI Tools: Copyleaks / Turnitin (institutional access); DeepL Translator (free); Wix (free); Adobe Portfolio (free for students); ChatGPT / Gemini (free tiers)

Text Books:

1. Raman M. and Sharma S., Technical Communication: Principles and Practice, Oxford University Press, 4th Edition, 2022.
2. Shetty J. and Pareek R., Communicative English for Engineers and Professionals, Wiley India, 3rd Edition, 2021.

Reference Books:

1. Rizvi M.A., Effective Technical Communication, Tata McGraw-Hill, 2nd Edition, 2018.
2. Murphy H.A., Effective Business Communication, McGraw-Hill Education, 7th Edition, 2019.
3. Wren P.C. and Martin H., High School English Grammar and Composition, S. Chand, Revised Edition, 2019.

Web Resources & NPTEL Links:

1. British Council – Learn English – https://learnenglish.britishcouncil.org
2. NPTEL – Communication Skills – https://nptel.ac.in/courses/109104101
3. Purdue OWL (Online Writing Lab) – https://owl.purdue.edu
4. BBC Learning English – https://www.bbc.co.uk/learningenglish

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES05101T	ES	Computational Thinking & Problem Solving with Python (Common to all branches of Engineering)	3	0	0	0	3

Pre-Requisites: Basic knowledge of computer fundamentals, logical reasoning, problem-solving, and elementary mathematics.

Course Objectives:

- To introduce students to computational thinking and problem-solving techniques.
- To develop the ability to design algorithms, flowcharts, and pseudocode.
- To provide foundational knowledge of Python programming — data types, operators, and I/O.
- To enable students to implement logic using control structures and data structures.
- To develop skills in modular programming, file handling, and basic OOP concepts.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply computational thinking concepts to design algorithms, flowcharts, and basic Python programs. (L3)

CO2: Develop programs using decision-making and looping constructs to solve logical problems. (L3)

CO3: Analyse and manipulate data using strings, lists, tuples, and sets for problem solving. (L4)

CO4: Design modular programs using functions, recursion, and dictionaries to improve code reusability. (L4)

CO5: Implement file handling, exception handling, and basic object-oriented programming concepts in Python applications. (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	3		1				2	3	3
CO2	3	3	3	2	3		1				2	3	3
CO3	3	3	3	2	3		1				2	3	3
CO4	3	3	3	2	3		1				3	3	3
CO5	3	3	2	3	3		1				3	3	3

Unit I: Computational Thinking and Programming Basics:

Computational Thinking: Characteristics, problem-solving strategies, steps in problem solving; algorithms — definition and properties; flowcharts — symbols and construction; pseudocode — writing and conversion; abstraction, decomposition, pattern recognition, algorithm efficiency basics.

Introduction to Python: Installation and execution environment; variables, identifiers, keywords; data types, type conversion; input and output statements; expressions and operators — arithmetic, relational, logical, assignment; operator precedence.

AI Tools Integration: Use Flowgorithm to design and simulate algorithms (e.g., largest of 3 numbers); draw flowcharts using Lucidchart; use Algorithm Visualizer to trace algorithm execution; use ChatGPT to convert pseudocode to Python and verify output.

Unit II: Decision Making and Looping:

Decision Control Statements: Boolean expressions; if, if-else, if-elif-else, nested if; conditional

expressions (ternary operator).

Looping Statements: while loop, for loop, iteration techniques, nested loops, infinite loops; loop control — break, continue, pass; else with loops.

Practical Problem Solving: prime number check, pattern programs using nested loops, menu-driven programs using control statements.

AI Tools Integration: Use Python Tutor to visualize if-else and loop execution step by step; use Thonny to debug loop-based programs; use Replit for online coding practice; use ChatGPT to generate test cases for control flow programs.

Unit III: Strings and Data Structures:

Strings: Representation, indexing, slicing, operations, built-in functions and methods.

Lists: Creation, indexing and slicing, operations, functions, methods, nested lists.

Tuples: Creation, operations, packing and unpacking.

Sets: Creation, set operations — union, intersection, difference; frozen sets.

AI Tools Integration: Use NLTK and TextBlob to perform text analysis (sentiment, word frequency) on strings; use Pandas and NumPy for list and array operations; use ChatGPT to explain string slicing and list comprehensions with concrete examples.

Unit IV: Functions and Problem Solving:

Dictionaries: Creation, operations, methods; dictionary-based applications.

Functions: Built-in and user-defined functions; function definition and calling; arguments — positional, keyword, default, variable-length; scope of variables (local and global).

Recursion: Recursive functions — factorial, Fibonacci; lambda functions (anonymous functions); applications of functions in problem solving.

AI Tools Integration: Use Jupyter Notebook for interactive function experiments; use SymPy for symbolic mathematical expressions; use PyCharm for modular program development; use ChatGPT to trace recursive function calls step by step.

Unit V: File Handling, Exception Handling, and OOP:

Modules and Packages: Creating and importing modules; standard library modules.

File Handling: Opening, reading, writing, and closing files; file modes; working with text files; processing CSV and Excel files.

Exception Handling: Types of errors (syntax, runtime, logical); try, except, finally blocks; raising exceptions.

Introduction to OOP: Classes, objects, attributes, methods, constructors, self-keyword; basic applications of OOP in Python.

AI Tools Integration: Use Pandas and OpenPyXL to read and write CSV/Excel files; use Pytest for testing exception handling; use ChatGPT to explain OOP class design with real-world examples; use Django for a simple mini web application as a capstone project.

Text Books:

1. Pieter Spronck, Computational Thinking with Python, (course textbook — covers computational thinking, algorithms, and Python programming).
2. Reema Thareja, Programming in Python, Oxford University Press. (Primary implementation reference — all five units.)

Reference Books:

1. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, 2nd Edition, O'Reilly Media. (Free at greenteapress.com)
2. Eric Matthes, Python Crash Course, 2nd Edition, No Starch Press.

Online Tools Used in this Course:

1. Algorithm & Flowchart: Flowgorithm – flowgorithm.org |Lucidchart – lucidchart.com | Algorithm Visualizer – algorithm-visualizer.org
2. Programming & Debugging: Python Tutor – pythontutor.com | Thonny – thonny.org |Replit – replit.com
3. Data & Scientific: Pandas – pandas.pydata.org | NumPy – numpy.org |SymPy – sympy.org
4. NLP & Text: NLTK – nltk.org |TextBlob – textblob.readthedocs.io
5. Development: Jupyter Notebook – jupyter.org | PyCharm – jetbrains.com/pycharm |OpenPyXL – openpyxl.readthedocs.io
6. NPTEL – Problem Solving through Programming in C / Python – nptel.ac.in

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES03103T	ES	**Design Thinking** (Common for all branches of Engineering)	1	0	4	0	3

Pre-Requisites: Basic knowledge of problem-solving, creativity, communication, teamwork, and fundamental engineering concepts.

Course Objectives:

- To understand the principles, philosophy, stages, and importance of Design Thinking in human-centred engineering problem solving.
- To apply empathy methods such as interviews, observation, personas, and problem-framing techniques to identify and understand user needs.
- To analyse user needs, formulate effective problem statements using the How Might We framework, and leverage AI tools for generating insights and evaluating solutions.
- To develop, test, evaluate, and pitch innovative prototype solutions using AI ideation tools, user feedback, and digital prototyping platforms.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Recall and understand the philosophy, principles, and stages of the Design Thinking process and its relevance to human-centred problem solving. (L3)

CO2: Apply empathy tools (interviews, observation, personas) and problem-framing techniques to real-world engineering scenarios. (L3)

CO3: Analyse user needs and problem statements using the 'How Might We' framework and AI-powered insight generation tools. (L4)

CO4: Evaluate and select the most viable prototype solution through structured testing, user feedback, and AI-assisted evaluation rubrics. (L5)

CO5: Create innovative prototypes and pitch solutions to identified problems using AI ideation tools and digital prototyping platforms. (L6)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	2	2	1	1	2	1				2	1	1
CO2	2	2	3	2	2	3	1				2	2	1
CO3	3	3	3	2	3	3	1				3	2	1
CO4	3	3	3	3	3	2	2				3	2	2
CO5	3	3	3	3	3	2	2				3	2	2

Unit 1: Introduction to Design Thinking:

Core Concepts:

- Design Thinking: Definition, origin (Stanford d.school, IDEO), and relevance to engineering
- The 5-Stage Model: Empathise – Define – Ideate – Prototype – Test
- Human-centred design principles: Desirability, Feasibility, Viability
- Systems Thinking vs. Linear Thinking: Understanding complexity
- Case Studies: Successful design thinking applications (IDEO shopping cart, Stanford Mobile Clinic)

AI Integration:

- AI Tool: ChatGPT / Copilot – use AI to explain design thinking in different domains (healthcare, education, agriculture)

- Activity: Use MindMeister (free, open-source alternative: FreeMind) to create a mindmap of the 5-stage process
- AI Video Analysis: Watch an IDEO documentary and use AI to summarise key DT principles observed

Unit II: Empathise and Define Stages:

Core Concepts:

- Empathy Methods: Interviews, Observation, Immersion, Shadowing
- Creating User Personas: Demographic, psychographic, goals and pain points
- Empathy Mapping: Says, Thinks, Does, Feels
- Point of View (POV) Statement: User + Need + Insight formula
- How Might We (HMW) Questions: Reframing problems as opportunities

AI Integration:

- AI Tool: Miro (free plan) – create digital empathy maps and POV statements collaboratively
- AI Tool: ChatGPT – generate diverse user persona profiles for a given problem context
- Activity: Interview 5 peers about a campus problem; use AI to analyse interview transcripts and extract themes
- AI Tool: Dovetail (free plan) – affinity mapping of qualitative research data

Unit III: Ideation Stage:

Core Concepts:

- Creative Thinking Techniques: Brainstorming, Brainwriting, SCAMPER, Random Word Association
- Lateral Thinking: Edward de Bono's Six Thinking Hats
- Affinity Diagramming: Grouping ideas by themes
- Prioritisation Matrix: Impact vs. Effort, Dot Voting
- Concept Selection: Morphological charts and Pugh Matrix

AI Integration:

- AI Tool: ChatGPT – use 'expand my idea' prompts to generate 20 variations of a concept
- AI Tool: IdeaFlip (free) / Miro – digital brainstorming and affinity diagramming
- AI Tool: Midjourney (free trial) / DALL-E (free credits) – generate visual concept sketches from text descriptions
- Activity: Run a 30-minute AI-augmented brainstorming session and compare idea quality with pure human brainstorming

Unit IV: Prototyping:

Core Concepts:

- Principles of Prototyping: Low-fidelity vs. High-fidelity prototypes
- Paper Prototyping: Sketching UI and physical models
- Digital Wireframing: Introduction to wireframes and mockups
- 3D Prototyping Concepts: Introduction to 3D printing and rapid prototyping
- Role Plays and Experience Prototyping: Service design scenarios

AI Integration:

- AI Tool: Figma (free plan) – create digital wireframes and interactive mockups
- AI Tool: Uizard (free plan, AI-powered) – convert hand-drawn sketches to digital wireframes using AI
- AI Tool: Tinkercad (free, web-based) – create simple 3D models for physical prototyping
- Activity: Build a paper prototype for a chosen problem and photograph for digital documentation

Unit V: Testing, Iteration and AI in Design:

Core Concepts:

- User Testing Methods: Think-aloud protocol, A/B testing, Usability testing
- Feedback Analysis: Affinity mapping of test feedback
- Iteration Cycle: Incorporating feedback and pivoting
- Design Presentation: Pitching to stakeholders (elevator pitch, demo day format)
- AI in Design: Overview of AI-assisted design (generative design, co-creation with AI)

AI Integration:

- AI Tool: Maze (free plan) – conduct remote usability tests on Figma prototypes
- AI Tool: ChatGPT – analyse collected user feedback and generate insights and improvement suggestions
- Activity: Present a final prototype to a panel; use AI to prepare pitch slides in Canva
- Reflection: Create a Design Thinking journal documenting each stage with AI tool usage notes

Text Books:

1. Brown, Tim. (2019). Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. HarperBusiness, New York.
2. Lewrick, Michael, Link, Patrick & Leifer, Larry. (2018). The Design Thinking Playbook. Wiley, New Jersey.

Reference Books:

1. IDEO.org. (2015). The Field Guide to Human-Centered Design. IDEO.org, San Francisco.
2. Kelley, Tom & Kelley, David. (2013). Creative Confidence: Unleashing the Creative Potential Within Us All. Crown Business.
3. Liedtka, Jeanne & Ogilvie, Tim. (2011). Designing for Growth: A Design Thinking Tool Kit for Managers. Columbia Business School Press.
4. Plattner, Hasso, Meinel, Christoph & Leifer, Larry (Eds.). (2018). Design Thinking Research. Springer.
5. Kumar, Vijay. (2012). 101 Design Methods: A Structured Approach for Driving Innovation. Wiley.
6. Stickdorn, Marc & Schneider, Jakob. (2021). This Is Service Design Thinking (Expanded ed.). BIS Publishers.

Web Resources:

1. Stanford d.school Resources: dschool.stanford.edu/resources
2. IDEO Design Kit: www.designkit.org/methods
3. MIT OpenCourseWare – Design Thinking: ocw.mit.edu
4. Interaction Design Foundation: www.interaction-design.org
5. NPTEL Design Thinking: swayam.gov.in

YouTube / Video Resources:

1. YouTube: Stanford d.school – Design Thinking Bootcamp – www.youtube.com/@stanforddschool
2. YouTube: IDEO U – www.youtube.com/@ideou
3. YouTube: Crash Course – Design Thinking Series – www.youtube.com/@crashcourse
4. YouTube: AJ&Smart – Design Sprint tutorials – www.youtube.com/@AJSmart
5. YouTube: NPTEL Design Thinking – www.youtube.com/c/nptel

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26BSHB110P	BS	Applied Chemistry for Computer Science Lab (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	0	0	0	3	1.5

Pre-Requisites: Applied Chemistry for Computer Science (co-requisite).

Course Objectives:

- To perform classical wet chemical analyses (redox, acid-base) on industrially relevant systems including battery electrolytes and iron solutions.
- To determine corrosion rates and understand the influence of environmental factors using standard electrochemical and gravimetric methods.
- To apply instrumental methods—conductometry, potentiometry, spectrophotometry, and IR spectrophotometry—for quantitative and qualitative chemical analysis.
- To synthesise a polymer (Bakelite) and relate its properties to structure and industrial relevance for electronics applications.
- To automate experimental data analysis using Python and develop AI-based models for corrosion prediction and spectral analysis.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Perform redox and conductometric titrations to estimate concentrations of analytes in solution. (L3)

CO2: Determine corrosion rates and analyze environmental factors affecting corrosion of metals. (L3)

CO3: Apply potentiometric and spectrophotometric techniques for quantitative chemical analysis. (L3)

CO4: Identify functional groups in organic compounds using IR spectroscopy. (L4)

CO5: Synthesize a polymer and use AI/Python-based data analysis for predictive modelling of material behaviour. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	3	3	1	3	3	3		2	2	1
CO2	3	3	2	3	3	3	3	3	3		2	2	1
CO3	3	3	2	3	2	3	3	3	3		2	2	2
CO4	3	3	3	3	3	1	3	3	3		3	2	1
CO5	3	3	3	3	3	2	3	3	3		3	2	2

List of Experiments:

Regular Experiments:

1. Estimation of Ferrous Iron (Fe^{2+}) using Potassium Dichromate ($K_2Cr_2O_7$) – Redox Titrations
2. Determination of Strength of Acid in Lead Acid Battery
3. Determination of Corrosion Rate (Weight Loss Method)
4. Determination of Corrosion by Environment (Effect of pH, Salt Concentration, and Temperature)

Instrumental Methods of Chemical Analysis:

5. Conductometric Titration of Strong Acid vs. Strong Base (HCl vs. NaOH)
6. Conductometric Titration of Weak Acid vs. Strong Base (CH_3COOH vs. NaOH)
7. Potentiometry (Redox Titration) – Estimation of Iron (Fe^{2+}) using Potassium Dichromate ($K_2Cr_2O_7$)

8. Preparation of a Polymer (Bakelite)
9. Spectrophotometry – Determination of Wavelength (λ) and Verification of Beer–Lambert Law
10. Identification of Functional Groups in Organic Compounds by IR Spectrophotometer

AI Integration:

Data analysis using Python (NumPy, Pandas, Matplotlib) for titration curve plotting and equivalence point detection. Predictive modelling for corrosion rate and material behaviour using scikit-learn regression. TensorFlow for optional IR spectrum pattern classification. OpenCV for colorimetric image-based concentration estimation. Qiskit (introductory) for exploring quantum chemistry concepts relevant to spectroscopy.

Free / Open-Source AI Tools: Python (NumPy, Matplotlib, scikit-learn) – Google Colab (free); LibreOffice Calc (free) – data recording; Virtual Chemistry Lab – VLabs IIT (free)

Text Books:

1. Jain, P. C., & Jain, M. (2020). Engineering Chemistry (17th ed.). Dhanpat Rai Publishing.
2. Shika Agarwal (2026). Engineering Chemistry (5th ed.). Cambridge University Press**.**

Reference Books:

1. Dara, S. S., & Umare, S. S. (2018). Engineering Chemistry (15th ed.). S. Chand Publishing.

Online Resources:

1. IEEE Xplore Digital Library
2. Semiconductor Industry Association resources
3. NPTEL Engineering Chemistry
4. ASME technical resources

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26HMHS101P	HM	Communicative English Lab (Common for all branches of Engineering)	0	0	0	2	1

Pre-Requisites: Basic knowledge of English grammar, vocabulary, reading, writing, and everyday communication, with a willingness to participate in listening, speaking, and digital language-learning activities.

Course Objectives:

- To develop students' oral communication skills through structured speaking and listening exercises.
- To enhance pronunciation, intonation, and fluency using AI-assisted speech tools.
- To build professional communication competencies including group discussion, debate, and presentation skills.
- To enable students to write effective professional documents such as emails, reports, and cover letters.
- To integrate AI tools in language learning for self-assessment, feedback, and continuous improvement

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Pronounce English words correctly using standard phonemic symbols and demonstrate improved speech fluency. (L3)

CO2: Apply active listening skills to comprehend spoken English and respond appropriately in formal and informal communication contexts. (L3)

CO3: Participate effectively in group discussions, presentations, and debates demonstrating logical argumentation. (L4)

CO4: Write grammatically correct and contextually appropriate professional documents including emails, reports, and resumes. (L6)

CO5: Utilize AI-powered tools for language learning, self-evaluation, and continuous improvement of English skills. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1						2	3	3	3		2		
CO2						2	3	3	3		2		
CO3						3	3	3	3		2		
CO4						2	3	3	3		2		
CO5						2	3	3	3		2		

Lab Syllabus – 12 Topics with AI Integration:

S. No.	Lab Topic	Lab Activities	AI Activity	AI Tools	COs
1.	Phonetics & Pronunciation Practice	Record & playback vowel/consonant articulation; IPA chart drill; Minimal pair exercises	Use AI speech-to-text to detect mispronunciations; real-time phoneme feedback via ELSA Speak	ELSA Speak, Google Speech Recognition, Speechling	CO1 & CO5
2.	Listening	Listen to TED	Auto-generate	Otter.ai,	CO2

	Comprehension &Note-Taking	Talks/podcasts; fill-in-the-blank exercises; summarise audio passages	transcripts; compare student notes with AI-generated summaries using Otter.ai	YouTube Auto-captions, Whisper AI	& CO5
3.	Spoken English & Fluency Building	Impromptu speaking (1–2 min); picture description; story completion tasks	AI analyses speech for filler words, pace, and clarity; ChatGPT prompts for spontaneous speaking	Speeko, ChatGPT, Google Assistant	CO1, CO3 & CO5
4.	Group Discussion Techniques	Role-based GD on current topics; turn-taking practice; consensus building	AI evaluates argument structure and language; ChatGPT generates counter-arguments for practice	ChatGPT, Gemini, Mentimeter	CO2 & CO3
5.	Public Speaking & Presentations	Prepare and deliver 3-minute presentations; use of visual aids; Q&A handling	AI slide deck generation; real-time delivery feedback on pacing and confidence	Beautiful.ai, Gamma.app, Microsoft Copilot	CO3 & CO5
6.	Interview Skills & Mock Interviews	HR round simulation; STAR method answers; body language and eye contact	AI-powered mock interview with instant feedback on content, grammar, and tone	Yoodli, Interview Warmup by Google, ChatGPT	CO1, CO3 & CO5
7.	Professional Email & Business Writing	Draft formal emails, complaints, and requests; subject line formulation; netiquette	AI refines email drafts; tone & clarity check; compare student draft with AI-improved version	Grammarly, ChatGPT, Gemini	CO4 & CO5
8.	Resume & Cover Letter Writing	ATS-friendly resume design; action verbs usage; tailoring cover letters to job descriptions	AI resume scanner; keyword optimisation suggestions; ChatGPT cover letter generation and critique	Kickresume, Resume.io, Jobscan, ChatGPT	CO4 & CO5
9.	Reading Comprehension & Critical Thinking	Read articles/editorials; identify main idea, inference, and tone; speed reading drills	AI generates comprehension questions; text summarisation and comparative analysis	Quillbot, Speechify, NotebookLM	CO2 & CO4
10.	Debate & Argumentation Skills	Structured debate (for/against); Rebuttal techniques; logical fallacy identification	AI generates arguments on both sides; evaluate logical structure using ChatGPT	ChatGPT, Claude.ai, Kialo Edu	CO2 & CO3
11.	Report Writing & Technical Communic	Lab report, incident report, and project report formats; headings, data presentation	AI improves report coherence and passive/active voice usage; grammar and	Grammarly Business, ChatGPT, Hemingway	CO4 & CO5

	ation		structure feedback	Editor	
12.	Etiquette, Netiquette & Personal Branding	Workplace conversation role play; LinkedIn profile creation; professional social media etiquette	AI reviews LinkedIn summaries; digital persona analysis; personal brand statement via ChatGPT	ChatGPT, Canva AI, LinkedIn AI features	CO3, CO4 & CO5

Text Books:

1. Raman, Meenakshi & Sharma, Sangeeta. Technical Communication: Principles and Practice (3rd Ed.). Oxford University Press, 2022.
2. Krishnaswamy, N. & Krishnaswamy, L. The Story of English in India. Foundation Books, Cambridge University Press, 2021.

Reference Books:

1. Swan, Michael. Practical English Usage (4th Ed.). Oxford University Press, 2016.
2. Murphy, Raymond. English Grammar in Use (5th Ed.). Cambridge University Press, 2019.
3. Rizvi, M. Ashraf. Effective Technical Communication (2nd Ed.). McGraw-Hill Education, 2018.
4. Turton, N.D. & Heaton, J.B. Longman Dictionary of Common Errors. Pearson Longman, 2017.
5. Lesikar, Raymond V. & Flatley, Marie E. Basic Business Communication (12th Ed.). McGraw-Hill, 2020.
6. Rutherford, Andrea J. Basic Communication Skills for Technology (3rd Ed.). Pearson, 2019.

Online Resources

1. BBC Learning English – https://www.bbc.co.uk/learningenglish (Grammar, pronunciation, and listening)
2. British Council Learn English – https://learnenglish.britishcouncil.org
3. TED Talks – https://www.ted.com/talks (Listening & speaking practice)
4. Coursera English Communication Skills Specialisation – https://www.coursera.org
5. edX English for Professional and Academic Purposes – https://www.edx.org
6. Grammarly Blog – https://www.grammarly.com/blog (Writing tips)
7. ELSA Speak (App) – https://elsaspeak.com (AI pronunciation coach)

YouTube Channels:

1. English with Lucy – @EnglishwithLucy (Vocabulary, speaking, British accent)
2. Learn English with TV Series – @LearnEnglishWithTVSeries
3. Rachel's English – @rachelsenglish (American accent and pronunciation)
4. TED-Ed – @TEDEd (Critical thinking and communication ideas)
5. Spoken English Hub – @SpokenEnglishHub (Indian context, interview prep)
6. JenniferESL – @JenniferESL (Grammar and business English)

NPTEL / Swayam Courses:

1. Developing Soft Skills and Personality, NPTEL / SWAYAM, IIT Kanpur, https://swayam.gov.in

2. Business English Communication Suite, Coursera (Free Audit), University of Washington, https://coursera.org
3. Speak English Professionally: In Person, Online & On the Phone, Coursera, Georgia Tech, https://coursera.org
4. English and Academic Preparation, NPTEL, IIT Bombay, https://nptel.ac.in
5. Technical English for Engineers, NPTEL / SWAYAM, IIT Roorkee, https://swayam.gov.in

Lab Software & AI Tools Required:

Software / Tool	Type	Purpose
ELSA Speak	AI Pronunciation Coach App	Real-time speech accuracy & fluency feedback
Grammarly	AI Writing Assistant	Grammar, punctuation, style correction
ChatGPT / Claude.ai	Generative AI Chatbot	Conversation practice, writing assistance, debate prep
Otter.ai	AI Transcription Tool	Transcribe and analyse spoken content
Yoodli	AI Public Speaking Coach	Presentation and interview delivery analysis
Hemingway Editor	Writing Clarity Tool	Improve readability and sentence structure
Mentimeter	Interactive Presentation	Live polls, word clouds during GD sessions
Audacity	Audio Recording Software	Record and review pronunciation exercises
Google Meet / Zoom	Video Conferencing	Mock interviews and virtual presentations
Canva / Beautiful.ai	AI Presentation Design	Create professional visual presentations

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES05101P	ES	Computational Thinking & Problem Solving with Python Lab (Common to all branches of Engineering)	0	0	0	3	1.5

Pre-Requisites: Basic computer literacy, logical reasoning, problem-solving skills, and elementary mathematics.

Course Objectives:

- To develop problem-solving skills through algorithms, flowcharts, and basic Python programming.
- To enable students to implement decision-making and iterative solutions using Python control structures.
- To develop programming skills for manipulating strings and built-in data structures such as lists, tuples, and sets.
- To equip students with modular programming, file handling, exception handling, and basic object-oriented programming skills.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply algorithms, flowcharts, variables, data types, and operators to design and implement basic Python programs for solving simple computational problems. (L3)

CO2: Apply conditional and iterative control structures, including if-else, nested if, match, while, and for loops, to develop solutions for decision-making and repetitive problems. (L3)

CO3: Analyze string operations and Python data structures such as lists, tuples, and sets to manipulate, organize, and solve data-oriented problems. (L4)

CO4: Evaluate computational problems and develop modular Python solutions using user-defined functions, recursion, lambda expressions, and dictionaries. (L5)

CO5: Apply file handling, exception handling, and basic object-oriented programming concepts to develop reliable and maintainable Python applications. (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	1	3		3	3	3		2	3	3
CO2	3	3	3	2	3		3	3	3		2	3	3
CO3	3	3	3	2	3		3	3	3		2	3	3
CO4	3	3	3	2	3		3	3	3		3	3	3
CO5	3	3	3	3	3		3	3	3		3	3	3

List of Experiments:

Unit I: Computational Thinking and Programming Basics:

1. **Algorithm Design, Flowcharts, and Python Basics:**

   Draw flowcharts using Flowgorithm for three problems: (a) find the largest of two numbers; (b) check whether a number is even or odd; (c) compute the factorial of a given number. For each flowchart, trace through the logic manually with 3 sample inputs before coding. Write equivalent Python programs for all three; run them in Thonny or Google Colab and verify that the output matches the manual trace. Use Python Tutor to visualize the variable state at each step for the factorial program.

2. Data Types, Operators, and Type Conversion:

Write Python programs to: (a) declare variables of all primitive data types (int, float, complex, bool, str) and print their type using type(); (b) demonstrate all operator categories — arithmetic (+, −, ×, ÷, //, %, **), relational (==, !=, <, >, <=, >=), logical (and, or, not), bitwise (&, |, ^, ~, <<, >>), and assignment operators — with one example each; (c) demonstrate implicit and explicit type conversion — int to float, float to int, int to str, str to int using int(), float(), str(); (d) print a formatted bill: accept item name, quantity, and price as input; compute and display total using f-string formatting with two decimal places.

3. Input, Output, and Expression Evaluation:

Write Python programs to: (a) accept the radius of a circle from the user; compute and display area (πr^2) and circumference ($2\pi r$) using the math module; (b) accept three subject marks; compute total, percentage, and grade (O ≥ 90, A+ ≥ 80, A ≥ 70, B+ ≥ 60, B ≥ 50, C ≥ 40, F < 40); print a formatted result card using print() with appropriate spacing; (c) accept two numbers and an operator (+, −, ×, ÷) as input; evaluate the expression using eval(); handle division by zero using a simple if condition; (d) demonstrate operator precedence: compute and print results of 5 mixed expressions — predict the result manually first, then verify by running the program.

Unit II: Decision Making and Looping:

4. Decision Control — if, elif, else, and match:

Write Python programs to: (a) accept a year and check whether it is a leap year using nested if — a year is a leap year if divisible by 400, or divisible by 4 but not by 100; test with 2000, 1900, 2024, 2023; (b) accept a character and classify it as uppercase letter, lowercase letter, digit, or special character using if-elif-else; (c) accept a day number (1–7) and print the day name using match-case (Python 3.10+); add a default case for invalid input; (d) implement a simple ATM menu using nested if — display options (Check Balance, Deposit, Withdraw, Exit); perform the selected operation; display appropriate error messages for invalid PIN or insufficient balance.

5. Loops — while, for, and Nested Loops:

Write Python programs to: (a) print multiplication table of a number entered by the user using a for loop (1 to 10); (b) print all prime numbers between 1 and 100 using a while loop with a flag variable — also count and display the total number of primes found; (c) print the following five patterns using nested for loops: right-angled triangle (stars), inverted triangle, number pyramid, Floyd's triangle, and a diamond pattern — accept the number of rows as input for each; (d) implement a number guessing game — generate a random number between 1 and 50 using random.randint(); allow the user a maximum of 7 attempts; print 'Too High' or 'Too Low' as hints; use break when the correct number is guessed; display attempts used.

6. Loop Control and Application Programs:

Write Python programs to: (a) compute the sum of digits, reverse, and check if palindrome for a number entered by the user — use a while loop extracting digits using % and //; test with 121, 12321, 1234; (b) compute the sum of the Fibonacci series up to N terms using a for loop; also print the series itself; (c) accept a positive integer N and compute: sum of natural numbers ($1+2+\ldots+N$), sum of squares ($1^2+2^2+\ldots+N^2$), and sum of cubes ($1^3+2^3+\ldots+N^3$) — verify sum of cubes formula: $[N(N+1)/2]^2$; (d) implement a simple calculator using a while loop with a menu — loop continues until the user selects Exit; use continue to handle invalid menu choices without breaking the loop.

Unit III: Strings and Data Structures:

7. String Operations and Methods:

Write Python programs to: (a) accept a string and perform: count vowels and consonants, reverse the string, check palindrome, count words, convert to title case — display all results; (b) demonstrate 10 built-in string methods: upper(), lower(), strip(), replace(), split(), join(), find(), count(), startswith(), endswith() — with one practical example each; (c) accept a sentence and: find the longest word, count frequency of each word (store in a dictionary), print words in alphabetical order; (d) implement a simple Caesar cipher — accept a message and a shift value (1–25); encrypt each letter by shifting it; display encrypted message; write a decrypt function to recover the original message.

8. Lists and Tuples:

Write Python programs to: (a) create a list of 10 integers; perform: insert at a given position, delete by value, find maximum and minimum without using built-in max/min, sort using Bubble Sort, search using linear search — display the list after each operation; (b) demonstrate list slicing: extract first half, second half, every alternate element, and reverse using slice notation; compare with string slicing on the same index values; (c) store student records as a list of tuples — each tuple: (roll_no, name, marks); sort by marks (descending) using sorted() with a lambda key; print rank list; find the student with highest marks using max() with a key; (d) demonstrate list comprehension: generate squares of even numbers from 1 to 20; filter words longer than 4 characters from a given word list; create a multiplication table as a 2D list using nested comprehension.

9. Sets and Applications:

Write Python programs to: (a) create two sets A and B from user input; compute and display: union (A | B), intersection (A & B), difference (A − B and B − A), symmetric difference (A ^ B); verify De Morgan's laws: not(A | B) == (not A) & (not B) for sample values; (b) use a set to remove duplicates from a list of 20 student roll numbers (with deliberate duplicates); compare list size before and after; print the unique roll numbers in sorted order using sorted(set()); (c) demonstrate frozen sets: create a frozenset of prime numbers up to 20; attempt to add an element (show TypeError); use it as a dictionary key (demonstrate immutability advantage); (d) implement a program to find: students who passed all three subjects, students who failed at least one, students who passed exactly two — store each class in a set and use set operations.

Unit IV: Functions and Problem Solving:

10. Dictionaries and Applications:

Write Python programs to: (a) create a student marks dictionary {name: [marks_list]}; compute and display average marks, highest scorer, and students who scored below 50; add a new student, update an existing student's marks, delete a student — show the dictionary after each operation; (b) implement a word frequency counter — accept a paragraph as input; split into words; build a frequency dictionary using a for loop; print the top 5 most frequent words sorted by count descending; (c) demonstrate all dictionary methods: keys(), values(), items(), get(), update(), pop(), setdefault(), copy() — with one example each; (d) build a simple phone book — store contacts as {name: phone}; support: add contact, search by name, delete contact, display all contacts sorted alphabetically — use a while loop menu.

11. User-Defined Functions and Scope:

Write Python programs to: (a) write 5 utility functions: is_prime(n), is_palindrome(s), factorial(n), gcd(a,b), power(base, exp) — call each with 3 test inputs and verify results; (b) demonstrate all four argument types: positional (def greet(name, age)), default (def greet(name,

age=18)), keyword (greet(age=20, name='Ram')), and variable-length (*args for sum of any count, **kwargs for student info display) — write one function for each and test; (c) demonstrate local vs. global scope: write a program with a global counter variable that is modified inside a function using the global keyword; contrast with a local variable of the same name — print both; (d) write a function find_stats(numbers) that accepts a list and returns mean, median, mode, and standard deviation without using the statistics module — test on [4, 7, 13, 2, 7, 4, 7, 1].

12. Recursion and Lambda Functions:

Write Python programs to: (a) implement recursive functions for: factorial(n), fibonacci(n), sum_of_digits(n), binary_search(arr, target, low, high), and power(base, exp) — for each, use Python Tutor to trace the call stack for a small input and count the total recursive calls; (b) compare iterative vs. recursive factorial for n = 5, 10, 15, 20 on execution time using time.time(); document when recursion becomes slower; (c) implement lambda functions for: square, cube, even/odd check, Celsius-to-Fahrenheit, and concatenate two strings — use each with map() and filter() on a list of 10 elements; (d) write a higher-order function apply_twice(f, x) that applies function f to x twice; test with square, double, and increment functions; demonstrate that lambda functions work as arguments.

Unit V: File Handling, Exception Handling, and OOP:

13. File Handling:

Write Python programs to: (a) create a text file and write 10 lines of student records (roll number, name, marks); read and display all lines; append 5 more records; count total lines; display only records where marks > 60; (b) copy the contents of one text file to another; count total characters, words, and lines in the source file; find and replace a specific word (e.g., 'Python' → 'programming') and write the result to a new file; (c) read a CSV file (student data) using the csv module; compute average marks per subject; write a summary report CSV with student name and grade; (d) create a student marks file; sort all records by marks in descending order (read all, sort in memory, rewrite); demonstrate the difference between read modes: 'r', 'w', 'a', 'r+' with examples.

14. Exception Handling:

Write Python programs to: (a) write a program that accepts two numbers and performs division; handle ZeroDivisionError, Value Error (non-numeric input), and Type Error in separate except blocks; use a finally block to print 'Operation complete' regardless of outcome; (b) implement a robust file reader that handles File Not Found Error and Permission Error; use try-with multiple except clauses; after handling each exception, print a meaningful user-friendly error message rather than a traceback; (c) define a custom exception class Insufficient Funds Error(Exception) with a message attribute; implement a Bank Account class with deposit() and withdraw() methods — withdraw() raises Insufficient Funds Error if balance is insufficient; test with deposits and withdrawals; (d) demonstrate exception chaining (raise X from Y) — catch a Value Error and re-raise as a custom Application Error with the original exception attached; use raise without arguments in a re-raise scenario.

15. Object-Oriented Programming — Class Design Capstone:

Design and implement a Student Management System using OOP: (a) create a Student class with attributes: roll_no, name, marks (list of 5 subjects) and methods: compute _percentage(), get_grade() (O/A+/A/B+/B/C/F), display() — create 5 Student objects with different data; (b) add a class variable total_students that increments with each object creation; add a class method get_total() and a static method validate_marks(marks) that returns True if all marks are between

0 and 100; (c) demonstrate object interactions: write a find_topper(students) function that accepts a list of Student objects and returns the one with the highest percentage; sort the list of Student objects by percentage using sorted() with a lambda key on obj.compute_percentage(); (d) write all student records to a file using the write() method; read them back and reconstruct Student objects; print the final class report in tabular format using f-strings with column alignment.

AI Tools Integration (Lab): Use Thonny IDE (beginner-friendly, built-in debugger) or Google Colab as the primary lab environment. Use Python Tutor (pythontutor.com) to visualize variable states, recursive call stacks, and list/dictionary operations — mandatory for Experiments 3, 12, and 15. Use Flowgorithm to design flowcharts before writing code in Experiments 1 and 4. Use ChatGPT or Claude to explain error messages when a program fails — read the explanation, fix the error yourself, and document both the error and the fix. Do NOT copy AI-generated code directly; use AI to understand concepts and debug, then write the solution independently.

Text Books:

1. Pieter Spronck, Computational Thinking with Python (course textbook — algorithms and Python foundations).
2. Reema Thareja, Programming in Python, Oxford University Press. (Primary implementation reference — all five units.)

Reference Books:

1. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, 2nd Edition, O'Reilly. (Free at greenteapress.com)
2. Eric Matthes, Python Crash Course, 3rd Edition, No Starch Press.

Online Tools and Resources:

1. Python Tutor – pythontutor.com |Flowgorithm – flowgorithm.org
2. Thonny IDE – thonny.org | Google Colab – colab.research.google.com
3. Python Official Documentation – docs.python.org/3
4. NPTEL – Problem Solving through Programming in Python – nptel.ac.in
5. W3Schools Python Tutorial – w3schools.com/python |Replit – replit.com

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26HMHS102L	HM	**Health and Wellness, Yoga and Sports** (Common for all branches of Engineering)	0	0	0	1	0.5

Pre-Requisites: Basic knowledge of health, fitness, nutrition, and hygiene; willingness to participate in yoga, meditation, sports, and physical activities; basic digital literacy; and an interest in using technology and AI tools for health, fitness, and wellness management.

Course Objectives:

- To create awareness about holistic health, wellness and preventive healthcare among students.
- To train students in yogic practices (asanas, pranayama, meditation) for physical and mental well-being.
- To enable students to design evidence-based personal fitness and nutrition plans.
- To develop sportsmanship, team spirit, fair play and physical fitness through indoor and outdoor sports.
- To integrate technology and AI tools for health monitoring, performance analysis and wellness management.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Demonstrate healthy lifestyle practices by applying principles of holistic health, balanced nutrition, physical fitness, stress management, sleep hygiene, personal hygiene, disease prevention, and ergonomic safety in daily life. (L3)

CO2: Demonstrate appropriate yoga, pranayama, meditation, mindfulness, and relaxation practices with correct techniques and breathing patterns to enhance flexibility, strength, balance, stress management, and overall well-being. (L3)

CO3: Analyze physical fitness components, training principles, sports skills, injury-prevention strategies, psychological factors, gender-inclusive practices, and sports technologies to evaluate and enhance individual and team performance. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1						3	3	3	2		2		
CO2						3	3	3	2		2		
CO3						3	3	3	2		2		

Unit I: Health, Wellness and Preventive Healthcare:

- Concept of health, wellness and fitness – definitions and seven dimensions
- WHO definition of health; holistic health model (Physical, Mental, Social, Spiritual)
- Nutrition basics – macronutrients, micronutrients, balanced diet, BMI calculation
- Lifestyle diseases – diabetes, hypertension, obesity – causes and prevention
- Mental health & stress management – causes, symptoms, COPE model
- Sleep hygiene and circadian rhythm regulation
- Substance abuse – awareness, prevention and rehabilitation
- Personal hygiene, sanitation and communicable disease prevention
- Ergonomics and occupational health for students

Unit II: Yoga, Pranayama, Meditation and Mind-Body Practice:

- Introduction to Yoga – history, philosophy, Patanjali's Ashtanga Yoga
- Surya Namaskar – 12-step sequence with correct breathing
- Asanas for flexibility: Tadasana, Trikonasana, Paschimottanasana, Sarvangasana
- Asanas for strength & balance: Virabhadrasana I & II, Vrikshasana, Naukasana
- Pranayama techniques: Anulom-Vilom, Bhastrika, Kapalbhati, Bhramari
- Meditation and mindfulness – Dharana, Dhyana, body-scan, guided relaxation
- Yoga Nidra for sleep and stress relief
- Yoga for specific conditions – stress, back pain, PCOD/PCOS, anxiety
- Shatkarmas (cleansing processes) – an overview; International Day of Yoga

Unit III: Physical Fitness, Sports Skills and Sports Technology:

- Physical fitness components – strength, endurance, flexibility, agility, speed, coordination
- Principles of training
- – overload, progression, specificity, reversibility, individuality
- Warm-up, cool-down and dynamic stretching routines
- Indoor sports: Badminton, Table Tennis, Chess, Carrom – rules and basic skills
- Outdoor sports: Cricket, Football, Volleyball, Kabaddi, Athletics – rules and skills
- Sports injuries – types, RICE method, prevention and return-to-play protocol
- Sports psychology – motivation, goal setting, concentration and visualisation
- Gender equity and inclusivity in sports; Para-sports awareness
- Sports technology – wearables, fitness trackers, GPS analytics, performance

Text Books:

1. Health and Physical Education, Dr. S. K. Mangal & Shubhra Mangal, PHI Learning Pvt. Ltd. 2021.
2. The Science of Yoga, William J. Broad, Simon & Schuster, 2012.

Reference Books:

1. Yoga: The Iyengar Way, Silva, Mira & Shyam Mehta, Dorling Kindersley, 2019.
2. Sports Medicine Essentials, Jim Clover, Cengage Learning, 2020.
3. Foundations of Physical Education & Sport, Charles A. Bucher, McGraw-Hill Education, 2018.
4. Mindfulness: An Eight-Week Plan for Finding Peace, Mark Williams & Danny Penman, Rodale Books, 2018.

Web Resources:

1. Ministry of AYUSH – https://main.ayush.gov.in
2. Yoga for Youth (International Day of Yoga) – https://yoga.ayush.gov.in
3. National Health Portal – https://nhp.gov.in
4. ICMR Dietary Guidelines – https://www.icmr.nic.in
5. Sports Authority of India – https://www.sportsauthorityofindia.nic.in
6. WHO Physical Activity Guidelines – https://www.who.int/news-room/fact-sheets/detail/physical-activity
7. Khelo India Programme – https://kheloindia.gov.in
8. NIMHANS (Mental Health Resources) – https://nimhans.ac.in

YouTube Resources:

1. Surya Namaskar – Complete 12-Step Tutorial – https://youtu.be/surya_tut_01

2. Pranayama for Beginners (Swami Ramdev) – https://youtu.be/pranayama_sr_02
3. Mindfulness Meditation (10 Min) – https://youtu.be/mindful_med_03
4. Balanced Nutrition Explained (ICMR) – https://youtu.be/nutrition_icmr_04
5. Sports Injury Prevention Tips – https://youtu.be/injury_prev_05
6. Mental Health Awareness for Students – https://youtu.be/mh_students_06
7. Yoga for Stress Relief – Isha Foundation – https://youtu.be/yoga_isha_07
8. Fit India Movement – PM Modi Launch – https://youtu.be/fitindia_pmo_08

AI-Integrated Activities:

S. No.	AI Activity	Description / Tool
1.	AI Fitness Assessment Dashboard	Log BMI, steps and sleep for 2 weeks using Google Fit / Apple Health; generate AI health-insight reports and compare with WHO benchmarks.
2.	Personalised Diet Planner with AI	Use NutriAI / Eat Right Now app to design a 7-day balanced meal plan; compare output with ICMR dietary guidelines.
3.	Yoga Pose Correction using ML Camera	Use YogaNotch / KAIA Health app for real-time AI posture feedback; record before/after improvement videos.
4.	Mental Health Chatbot Interaction & Reflection	Engage with Woebot / Wysa AI mental-health chatbot for one week; write a critical reflection on AI vs human counsellor effectiveness.
5.	Sports Performance Analytics with Python	Analyse a cricket/football dataset on Google Colab using Pandas; visualise player performance KPIs with Matplotlib / Seaborn.

B. Tech. – I Year I Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES05102A	MC	Cybersecurity Awareness for Engineers (Common for all branches of Engineering)	2	0	0	0	0

Pre-Requisites: Basic knowledge of computer systems, networking, internet usage, digital communication, information security, and fundamental cyber safety practices.

Course Objectives:

- To introduce students to the fundamentals of cybersecurity and secure digital practices in modern society.
- To create awareness about cyber threats, cybercrimes, and methods for protecting personal and institutional digital assets.
- To develop responsible digital citizenship through cyber ethics, privacy awareness, and legal understanding.
- To enable students to adopt cyber hygiene practices for secure use of digital technologies.
- To familiarize students with emerging cybersecurity concerns in cloud, digital platforms, and AI-enabled environments.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply fundamental cybersecurity concepts to address security requirements in digital environments. (L3)

CO2: Analyze common cyber threats, cyberattacks, and social engineering techniques to determine appropriate security measures. (L4)

CO3: Apply secure practices for protecting devices, networks, email, cloud services, and online platforms. (L3)

CO4: Evaluate cyber ethics, privacy protection measures, and applicable cyber laws in different digital scenarios. (L5)

CO5: Evaluate cybersecurity incidents and develop appropriate cyber hygiene and response strategies. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	3	2	2				2	3	3
CO2	3	3	3	3	3	2	2				3	3	3
CO3	3	3	3	2	3	2	2				3	3	3
CO4	2	3	2	3	2	3	3				3	3	2
CO5	3	3	3	3	3	3	3				3	3	3

Unit I: Introduction to Cyberspace and Cybersecurity:

Introduction to cyberspace and cybersecurity – Evolution of information systems and Internet – Importance of cybersecurity in engineering and society – Information security concepts – CIA Triad (Confidentiality, Integrity, Availability) – Digital footprint and digital identity – Security governance basics – Cyber threat actors – Engineering ethics and responsible digital behavior – Cyber hygiene practices.

Unit II: Cyber Threats, Cybercrime and Social Engineering:

Cyber threats and vulnerabilities – Cybercrime and cyber offenders – Malware: Virus, Worm, Trojan, Spyware, Ransomware – Cyberattack lifecycle and common attack methods – Phishing, Spear Phishing, Smishing and Vishing – Social engineering attacks – Password attacks and

credential theft – Identity theft and financial fraud – Insider threats – AI-enabled cyber threats and deepfakes – Case studies of major cyber incidents.

Unit III: Personal Cyber Safety and Cyber Security Technologies:
Password management and Multi-Factor Authentication – Authentication and access control concepts – Safe browsing practices – Email and communication security – Mobile device security – Cloud security basics – Secure use of Wi-Fi and public networks – Data backup and recovery – Cryptography and digital signatures (awareness level only) – Safe use of AI tools and digital platforms.

Unit IV: Privacy, Cyber Ethics and Cyber Laws:
Data protection and privacy concepts – Personal data and sensitive information – Cyber ethics and responsible digital behavior – Overview of cyber laws in India – Information Technology (IT) Act and related provisions (awareness level) – Common cybercrimes and legal implications – Cybercrime reporting mechanisms and grievance redressal – Digital evidence basics – Social media responsibility and legal considerations.

Unit V: Cyber Hygiene and Secure Digital Practices:
Cybersecurity risk awareness – Password management and Multi-Factor Authentication – Secure use of email, web, cloud and public networks – Safe digital transactions and digital payment awareness – Secure use of social media and AI tools – Cybersecurity awareness in IoT and emerging technologies – Incident awareness and cyber hygiene best practices.

Text Books:

1. Ajay Singh, Introduction to Cyber Security: Concepts, Principles, Technologies and Practices, Universities Press / Orient BlackSwan, Latest Edition.
2. Nina Godbole and SunitBelapure, Cyber Security: Understanding Cyber Crimes, Computer Forensics and Legal Perspectives, Wiley India.

Reference Books:

1. Charles J. Brooks, Christopher Grow, Philip Craig and Donald Short, Cybersecurity Essentials, Wiley
2. William Stallings and Lawrie Brown, Computer Security: Principles and Practice, Pearson.
3. Michael E. Whitman and Herbert J. Mattord, Principles of Information Security, Cengage Learning.
4. Joseph MiggaKizza, Guide to Computer Network Security, Springer.
5. Eric Cole, Cybersecurity for Beginners, Wiley.

Online Resources:

1. National Cyber Security Awareness Portal (India)
2. CERT-In (Indian Computer Emergency Response Team)
3. National Cyber Crime Reporting Portal
4. NIST Cybersecurity Framework Resources
5. OWASP Foundation
6. Cisco Networking Academy – Introduction to Cybersecurity

B. Tech. – I Year
II Semester

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26BSHB102T	BS	**Differential Equations & Vector Calculus** (Common for all branches of Engineering)	3	0	2	0	4

Pre-Requisites: Linear Algebra and Calculus

Course Objectives:

- To develop analytical and computational competence in solving ordinary and partial differential equations arising in engineering systems.
- To enable students to interpret vector differential and integral operators in physical and engineering contexts.
- To integrate mathematical modeling, visualization, and simulation using modern AI-assisted computational tools in alignment with NEP 2020.
- To apply ODE and PDE solution methods to model real-world engineering phenomena such as circuits, vibrations, heat transfer, and fluid flow.
- To use computational tools for solving, verifying, and visualizing differential equations and vector field problems.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Solve first-order ODEs and model engineering processes using analytical and computational approaches. (L3)

CO2: Analyse higher-order linear ODEs and interpret solutions in dynamic engineering systems such as circuits and oscillators. (L4)

CO3: Formulate and solve first-order and selected higher-order PDEs relevant to engineering models including wave, heat, and Laplace equations. (L3)

CO4: Interpret and compute gradient, divergence, and curl of scalar and vector fields and explain their physical significance. (L4)

CO5: Apply Green's, Stokes', and Divergence theorems to compute work, circulation, and flux and evaluate engineering field behaviour. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	1		1				2	3	2
CO2	3	3	3	2	2		1				2	3	2
CO3	3	2	2	2	1		1				2	3	2
CO4	3	3	3	3	2		1				2	3	2
CO5	3	3	2	3	3		1				2	3	2

Unit I: First-Order Ordinary Differential Equations:

Core Content: Classification of ODEs by order, degree, and linearity; linear first-order ODEs and integrating factor method; Bernoulli's equation; exact equations and integrating factors; orthogonal trajectories; Engineering applications of first-order ODEs; Newton's law of cooling and natural growth/decay models; electrical circuit models (RC and RL circuits); engineering applications of first-order ODEs.

AI Integration: Python SymPydsolve for symbolic ODE solution and verification; direction field (slope field) visualization using Matplotlib; AI-assisted debugging of integrating factor derivations; using ChatGPT/Copilot to explain solution families and physical interpretations; SciPy solve_ivp for numerical ODE solution with initial conditions.

Free/Open-Source AI Tools: Python (SymPy, SciPy, Matplotlib) on Google Colab; Jupyter Notebook; Wolfram Alpha free tier for solution verification; GNU Octave for ODE routines.

Unit II: Higher-Order Linear Differential Equations:

Core Content: Superposition principle; Wronskian and linear independence; complementary function for constant-coefficient ODEs (distinct real, repeated, and complex roots); particular integral by operator method; variation of parameters; simultaneous linear ODEs; Cauchy's (Euler) equation; Legendre's equation; applications to LCR circuits, simple harmonic motion, damped and forced oscillations; mass-spring-dashpot systems.

AI Integration: Python SciPy signal.lsim for simulation of underdamped, critically damped, and overdamped responses; resonance amplitude curve generation; AI-assisted Wronskian computation and verification using SymPy; using AI tools to explain transient vs. steady-state response; resonance and Q-factorinterpretation.

Free/Open-Source AI Tools: Python (NumPy, SciPy, SymPy, Matplotlib) on Google Colab; Jupyter Notebook; GNU Octave / MATLAB (optional) for LCR and mechanical system simulation; WolframAlpha free tier.

Unit III: Partial Differential Equations:

Core Content: Introduction and classification of PDEs; formation by elimination of arbitrary constants and functions; first-order linear PDEs and Lagrange's method; homogeneous linear PDEs with constant coefficients (CF and PI); standard forms of nonlinear first-order PDEs; Classification as hyperbolic, parabolic, and elliptic. Second-order PDE applications — wave equation, heat equation, and Laplace's equation by method of separation of variables.

AI Integration: Python SymPy for PDE formation and symbolic verification; 3D surface plot visualization of PDE solution families using Matplotlib; SciPy finite-difference simulation of the 1D heat equation with animation; AI-assisted PDE classification using the discriminant; prompting AI to classify mixed-derivative PDEs and student verification of AI reasoning.

Free/Open-Source AI Tools: Python (SymPy, NumPy, SciPy, Matplotlib) on Google Colab; Jupyter Notebook; Wolfram Alpha free tier for PDE classification verification; Scilab.

Unit IV: Vector Differentiation:

Core Content: Scalar and vector fields; del (∇) operator; gradient and directional derivative; unit normal to surfaces; angle between surfaces; divergence and physical interpretation; curl and physical interpretation; vector identities; conservative fields and scalar potentials; Laplacian; engineering applications of Gradient, Divergence, and Curl in electromagnetic and fluid mechanics contexts.

AI Integration: Python NumPy and Matplotlib for 2D and 3D gradient, divergence, and curl field visualization (quiver plots, level-surface overlays); SymPy for symbolic verification of vector identities; using AI tools to interpret Maxwell’s equations in terms of divergence and curl operators; AI-assisted explanation of irrotational and solenoidal field classification.

Free/Open-Source AI Tools: Python (NumPy, SymPy, Matplotlib, mpl_toolkits) on Google Colab; Jupyter Notebook; SciPy for numerical gradient computation; GNU Octave.

Unit V: Vector Integration and Integral Theorems:

Core Content: Line integrals over scalar and vector fields; work done and path independence; surface integrals and flux; Green's theorem in the plane and its applications; Stokes' theorem; Divergence theorem (Gauss's theorem); applications to area and volume computation; engineering applications in electromagnetism, fluid mechanics, and mass conservation.

AI Integration: Python SciPy integrate for numerical computation of line and surface integrals; 3D flux visualization with colour-coded normal flux using Matplotlib; numerical verification of Green's and Stokes' theorems; AI-assisted physical interpretation of the Divergence theorem in terms of source density; prompting AI to explain connections to Gauss's law and student evaluation of AI explanations.

Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib) on Google Colab; Jupyter Notebook; SymPy for symbolic integral verification; WolframAlpha free tier; 3D plotting tools.

Text Books:

1. Kreyszig E., Advanced Engineering Mathematics, 10th Edition, Wiley, 2018.
2. Grewal B.S., Higher Engineering Mathematics, 44th Edition, Khanna Publishers, 2017.
3. Thomas G.B., Weir M.D., Hass J., Thomas' Calculus, 14th Edition, Pearson, 2018.

Reference Books:

1. Zill D.G. and Wright W.S., Advanced Engineering Mathematics, 6th Edition, Jones & Bartlett, 2018.
2. Sneddon I.N., Elements of Partial Differential Equations, Dover Publications, 2006.
3. Arfken G.B. et al., Mathematical Methods for Physicists, 7th Edition, Academic Press, 2012.

Web Resources & NPTEL Links:

2. NPTEL: ODEs, PDEs, and Vector Calculus – https://nptel.ac.in
3. MIT OpenCourseWare – Differential Equations (18.03) and Multivariable Calculus (18.02): https://ocw.mit.edu
4. Python SymPy Documentation: https://docs.sympy.org
5. SciPy Documentation: https://docs.scipy.org

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26BSHB106T	BS	**Physics for Advanced Computing** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	2	1	0	0	3

Pre-Requisites: Engineering Mathematics, Basic Electronics, and Computer Science Fundamentals.

Course Objectives:

- To equip students with the concepts of interference, diffraction and polarization and their applications in optics and communication systems with basic AI-assisted analysis.
- To introduce lasers, fiber optics and advanced sensing technologies along with simple AI-assisted signal processing techniques.
- To provide basic knowledge of quantum mechanics and quantum computing with simple AI-assisted quantum applications.
- To familiarize students with semiconductors and magnetic materials and their applications using basic AI-assisted material analysis methods.
- To impart knowledge on low temperature physics and superconductivity and their applications with simple AI-assisted monitoring techniques.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply the principles of interference, diffraction and polarization to analyze optical systems and engineering applications. (L3)

CO2: Analyze the working of lasers, fiber optics and sensing systems with basic AI-assisted optical communication concepts. (L4)

CO3: Apply the concepts of quantum mechanics and quantum computing to explain modern computational and secure communication systems. (L3)

CO4: Analyze semiconductor characteristics and magnetic material behavior for engineering and electronic applications. (L4)

CO5: Apply the concepts of low-temperature physics and superconductivity in modern technological applications and intelligent systems. (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1	1	2		1				2	2	1
CO2	3	3	2	2	2		1				2	2	2
CO3	3	2	2	2	2		1				2	2	2
CO4	3	3	2	2	2		1				2	2	1
CO5	3	2	1	2	2		1				2	2	2

Unit I: Wave Optics and Quantum Information Processing:

Core Content: Interference: Principle of superposition and coherence; Interference; Types of interference; Interference in thin films (reflection geometry) and Newton's rings. Applications: Determination of wavelength and refractive index.

Diffraction: Introduction to diffraction; Fresnel and Fraunhofer diffraction; Fraunhofer diffraction due to single, double and N-slits (Diffraction Grating); Dispersive power and Resolving power of grating. Applications: CD/DVD data storage, Barcode scanners.

Polarization: Types of polarization; Nicol's prism; Half-wave and Quarter-wave plates.

AI Integration: AI-assisted polarization management for quantum key distribution. AI-integrated

holographic interferometry for surface flatness analysis.
Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib and scikit-learn).

Unit II: Quantum Photonics and Advanced Sensing:
Core Content: Lasers: Interaction of radiation with matter, Requisites and conditions for laser action, He-Ne laser, Nd-YAG laser, LiDAR basics. Laser-based data communication, Free-space optical communication, Optical data storage, Photonic computing basics, Applications of lasers.
Fiber Optics: Propagation mechanism, Numerical aperture, Acceptance angle, Types of optical fibres, Classifications and Modes of optical fibres, Attenuation and Losses in optical fibers, V-Number - Distributed Fiber Optic Sensing (DFOS) with AI-Powered Signal Analysis.
AI Integration: AI-powered signal analysis for distributed fiber optic sensing. Python simulation of laser propagation and fiber optic signal attenuation.
Free/Open-Source AI Tools: Google Colab, Python, NumPy, SciPy and Scikit-learn/ TensorFlow.

Unit III: Introduction to Quantum Mechanics & Quantum Computing:
Core Content: Introduction to quantum mechanics, de-Broglie concept (matter waves), Heisenberg's Uncertainty Principle, Wave function, Postulates of Quantum Mechanics, Schrödinger's time-independent wave equation, Particle in a one-dimensional potential well, Quantum tunnelling (basic idea).
Introduction to Quantum Computing: Classical bits vs Qubits. Concept of superposition and Entanglement, Basic Quantum Operations: Hadamard gate (idea of superposition), Pauli-X gate. Basic idea of quantum measurement.
Applications (qualitative only): AI-Assisted Tunnel Diodes and cold emission of electrons, Basic idea of quantum cryptography and secure communication.
AI Integration: AI-Assisted Analysis of Tunnel Diode Characteristics using Quantum Tunnelling Concepts, AI-Assisted Quantum Key Distribution for Secure Communication.
Free/Open-Source AI Tools: Google Colab, Python and Scikit-learn.

Unit IV: Semiconductors and Magnetic Materials:
Core Content: Semiconductors: Introduction; Fermi energy and Fermi level in intrinsic and extrinsic semiconductors; Expression for concentration of electrons in conduction band and holes in valence band; Electrical conductivity; Expressions for drift and diffusion currents; Hall effect; Expression for Hall coefficient and its applications.
Magnetism: Origin of magnetism; Dia, Para, Ferro, Ferri and Anti-ferro magnetic materials; Hysteresis; Soft and Hard magnetic materials. Applications (qualitative only): Magnetic data storage (MRAM), Hard Disk Drive and Magnetic shielding.
AI Integration: AI-Assisted Hall Effect Analysis for Semiconductor Characterization. AI-Assisted Magnetic Data Storage and MRAM Optimisation.
Free/Open-Source AI Tools: Google Colab, Python and Scikit-learn.

Unit V: Low Temperature Physics and Superconductivity:
Core Content: Introduction: Low temperature and its importance; Properties of materials at low temperatures; Liquefaction of gases (basic concept only); Liquid nitrogen and liquid helium (basic properties and uses). Applications (basic ideas): MRI, Rocket fuels, Cryogenic Cooling of High-Performance Computing System.
Superconductors: Definitions; Meissner effect; Type I and Type II superconductors; BCS theory: Cooper pairs and energy gap (qualitative); DC and AC Josephson effect. Applications (qualitative only): SQUIDs, magnetic levitation and lossless power transmission. Superconducting qubits: Charge, Flux, Phase.

AI Integration: AI-Assisted Cryogenic Cooling Optimization for High-Performance Computing Systems. AI-Assisted Stability Control in Superconducting Magnetic Levitation Systems.
Free/Open-Source AI Tools: Google Colab, Python and Scikit-learn.

Text Books:

1. Avadhanulu, Kshirsagar&Arun Murthy, A Textbook of Engineering Physics, S. Chand, 2019.
2. D. K. Bhattacharya & Poonam Tandon, Engineering Physics, Oxford Press, 2015.

Reference Books:

1. BPandey, B. K., & Chaturvedi, S. (2021). Engineering Physics. Cengage Learning.
2. Pillai, S. O. (n.d.). Solid State Physics. New Age International.
3. Nielsen, M. A., & Chuang, I. L. (n.d.). Quantum Computation & Quantum Information. Cambridge University Press.
4. Saleh, B. E. A., & Teich, M. C. (n.d.). Fundamentals of Photonics. Wiley.
5. Keiser, G. (n.d.). Optical Fiber Communications. McGraw Hill.
6. Tinkham, M. (n.d.). Introduction to Superconductivity. Dover Publications.

Web Resources & NPTEL Links:

1. NPTEL – nptel.ac.in
2. MIT OpenCourseWare – ocw.mit.edu
3. HyperPhysics – hyperphysics.phy-astr.gsu.edu
4. Feynman Lectures – feynmanlectures.caltech.edu
5. PhET Simulations – phet.colorado.edu

Recommended Free & Open-Source AI/Computational Tools:

1. Python (NumPy, SciPy, OpenCV, Matplotlib) – FREE on Google Colab for optics and EM simulations
2. IBM Qiskit (free, open-source) – For quantum computing and quantum cryptography simulations
3. scikit-learn (free, open-source) – For semiconductor data analysis and signal classification
4. OpenEMS (free, open-source) – For electromagnetic field simulation and antenna design
5. PhET Interactive Simulations (free) – For wave optics, semiconductors and quantum phenomena

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES03101T	ES	**Engineering Graphics** (Common to all branches of Engineering)	1	0	4	0	3

Pre-Requisites: Basic Geometry, and Fundamental Python Programming.

Course Objectives:

- To familiarise with the BIS conventions, geometric construction standards, and engineering scales.
- To develop visualization skills to interpret and project points, lines, and regular planes into 2D drawing sheets.
- To impart foundational knowledge on projecting regular 3D solids and reading directional geometric orientations.
- To enable the spatial reconstruction of sliced solids and flat unfolded developments of various structural shells.
- To introduce CAD tools combined with generative AI software to construct multi-view isometric representations from 3D models.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Construct engineering curves and scales using traditional drafting geometry validated by automated algorithmic paths. (L3)

CO2: Solve multi-planar projection instances of linear points, segments, and standard regular planar silhouettes. (L3)

CO3: Render projections of standard 3D solids and identify variations using structural boundary algorithms. (L4)

CO4: Draft complex cross-sectional geometries and mapped surface extensions of sliced geometric forms. (L4)

CO5: Produce composite isometric layouts alongside generative text-to-CAD system alternatives for dynamic prototyping. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1			2	1				1	2	2
CO2	3	3	2			2	1				1	2	1
CO3	3	3	2	1		2	1				1	2	2
CO4	3	3	3	1	2	2	1				2	2	2
CO5	3	2	2		3	2	1				2	2	2

Unit 1: Introduction to Engineering Graphics, Curves, and Scales:

Core Content:

Introduction: Principles of Engineering Graphics, signboards, BIS conventions, and lettering.

Curves: Construction of Polygons. Conic sections: Ellipse, Parabola, and Hyperbola using eccentricity/general methods. Cycloidal curves and Involutes.

Scales: Plain, Diagonal, and Vernier scales.

Hands-On Experience: Manual draft generation of an ellipse and diagonal scale, paired with a software verification loop.

AI Integration Module: AI prompt engineering for generating geometric profiles; algorithmic generation of conic curves using mathematical code blocks; parsing hand-drawn geometric sketches into clean vector shapes using Computer Vision (CV) edge detection (Canny Edge/Hough Transform).

Unit II: Orthographic Projections of Points, Lines:

Core Content: Principles of orthographic projections, projection of points in all four quadrants. Projection of straight lines inclined to one or both reference planes, traces of lines, determination of true lengths and true inclinations.

Projections of Planes: regular planes Perpendicular to both reference planes, parallel to one reference plane and inclined to the other reference plane; plane inclined to both the reference planes.

Hands-On Experience & Application:

Project: Construct a transformation script where a user inputting 2D projection endpoint automatically outputs the 3D line representation, true spatial length, and inclination angles without manual geometric tracking.

Application: Calibration setups for stereoscopic robotic arms tracking trajectories across multiple camera planes.

AI Integration Module: Machine learning pipelines for 3D coordinate estimation from 2D views; multi-view structural tracking; training neural models to predict true line length and orientation angles based solely on coordinate projections.

Unit III: Projection of Regular Solids:

Core Content:

Projection of Regular Solids: Types of solids: Polyhedra and Solids of revolution. Projections of solids in simple positions: Axis perpendicular to horizontal plane, Axis perpendicular to vertical plane and Axis parallel to both the reference planes, Projection of Solids with axis inclined to one reference plane and parallel to another plane.

Hands-On Experience & Application:

Project: Set up a classification notebook using scikit-learn or a simple convolutional model to automatically identify the type of solid (e.g., hexagonal prism vs. cone) present within an orthographic engineering layout image.

Application: Automated component sorting inside factory staging bays and conveyor streams.

AI Integration Module: Object recognition models (e.g., YOLO / Mobile Net) trained to identify standard solid geometries; automated multi view projection generation from 3D CAD meshes using programmatic render layers.

Unit IV: Sections of Solids, Development of Surfaces, and AI Generative Design:

Core Content:

Sections of Solids: Perpendicular and inclined section planes, Sectional views and True shape of section, Sections of solids in simple position only.

Development of Surfaces: Methods of Development: Parallel line development and radial line development. Development of a cube, prism, cylinder, pyramid and cone.

Hands-On Experience & Application:

Project: Use a nesting optimization script (e.g., using genetic algorithms or standard heuristics) to arrange several custom flat-surface solid developments onto a finite raw sheet grid, actively minimizing material margins and cut lines.

Application: Sheet metal stamping pattern automation in automotive and aerospace chassis manufacturing.

AI Integration Module: Generative Adversarial Networks (GANs) for generating flat-pattern configurations; algorithmic optimization of surface layouts to minimize sheet manufacturing scrap; predictive toolpath generation using physics-guided heuristic networks.

Unit V: Isometric Projections, Conversions, and 2D-to-3D Deep Reconstruction:
Core Content: Principles of isometric projection, isometric scale, isometric views/projections of planes, simple and compound solids. Conversion of isometric views to orthographic views and vice-versa.
Computer Graphics: Creating 2D&3D drawings of objects including PCB and Transformations using Auto CAD (Not for end examination).
Hands-On Experience & Application:
Project: Use a pretrained depth estimation or pixel-to-voxel neural model to parse a set of front, top, and side drawings, compiling them directly into an interactive 3D point cloud or boundary surface rendering.
Application: Instantly generating 3D printable mechanical files directly from archival 2D layout drafting logs.
AI Integration Module: 2D-to-3D depth reconstruction using Neural Radiance Fields (NeRF) or specialized generative models; auto-generation of 3D CAD models from simple multi-view engineering layout sketches.

Text Books:
1. Engineering Drawing by N.D. Bhatt, Charotar Publishing House 2016.
2. Engineering Drawing with an Introduction to AutoCAD, Dhananjay Jolhe, Tata McGraw Hill, 2017.

Reference Books:
1. Engineering Graphics and Design by Pradeep Jain, A.P. Gautam, Abhishek Bhargava, Khanna Publishing.
2. Engineering Drawing, K.L. Narayana and P. Kannaiah, Tata McGraw Hill, 2013.
3. Engineering Drawing, M.B.Shah and B.C. Rana, Pearson Education Inc,2009.

Video Links & Online Lectures:
1. **Classical Drafting Fundamentals**: NPTEL Engineering Graphical Design Course by IIT Kharagpur
2. **Computer Vision and Processing**: OpenCV Official Python Bootcamps and Tutorials
3. **Generative Design and Modeling Insights**: MIT 6.S192: Deep Generative Modeling Platforms.

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES31101T	ES	AI Tools and Applications (Common to all branches of Engineering)	1	0	4	0	3

Pre-Requisites: Basic computer literacy, internet usage, information-search skills, and fundamental communication and problem-solving skills.

Course Objectives:

- To introduce students to AI concepts and the landscape of AI tools available for everyday academic and professional use.
- To develop practical skills in using AI for document creation, presentations, writing, and academic tasks applicable across all engineering disciplines.
- To build proficiency in AI-powered data analysis, spreadsheet automation, and intelligent research tools without requiring programming knowledge.
- To expose students to AI tools in visual design, media creation, and note-taking, and progressively to AI applications specific to their engineering branch.
- To cultivate responsible AI habits: verifying outputs, protecting privacy, respecting intellectual property, and understanding ethical and legal boundaries of AI use.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply the fundamental concepts of Artificial Intelligence (AI) to explain its working principles and demonstrate the use of AI tools across text, image, audio, video, and code applications. (L3)

CO2: Use AI tools to create documents and presentations; apply prompt engineering and NotebookLM to improve writing, research, and study habits. (L4)

CO3: Perform data analysis using AI without coding; use AI-powered research and note-taking tools to find, verify, organize, and summarize information. (L4)

CO4: Apply AI tools in visual design, diagrams, and media; identify and verify AI applications relevant to their own engineering discipline. (L5)

CO5: Demonstrate responsible AI usage — bias, privacy, ethics, governance; design and present an AI-assisted mini-project using tools from all units. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	2	1	3	2	1				1	3	3
CO2	3	3	2	2	3	1	1				1	3	3
CO3	3	2	3	2	3	1	1				1	3	3
CO4	3	3	3	3	3	2	1				2	3	3
CO5	3	2	2	2	3	2	1				2	3	3

Unit I: Understanding AI and AI for Documents and Presentations:

What is AI: AI vs. ML vs. Deep Learning vs. Generative AI; brief history (Turing Test → ChatGPT → generative AI era); how LLMs work (training data, next-token prediction, temperature, hallucinations, context windows, knowledge cutoffs). AI ecosystem: ChatGPT, Claude, Gemini, Microsoft Copilot — strengths and differences.

AI for Presentations: Gamma.app — full slide deck from a paragraph prompt; Microsoft Copilot in PowerPoint — slide generation, design suggestions, speaker notes; Canva AI — Magic Design,

text-to-presentation, brand kits; Beautiful.ai — smart slide layouts; comparing AI-generated vs. manually created presentations.

AI for Document Writing: Microsoft Copilot in Word — draft, rewrite, summarize; Gemini in Google Docs — writing assistance and inline Q&A; Notion AI — meeting notes, project plans, knowledge base; Claude Projects — organizing documents and having a persistent AI assistant over a set of files.

AI Writing Assistance: Grammarly — grammar, tone, clarity, plagiarism; ChatGPT and Claude for lab reports, proposals, and letters; academic integrity — AI as an assistant not an author; disclosure norms; identifying AI-generated content using GPTZero.

AI Integration: Use Gamma.app to generate a presentation on an engineering topic in under 2 minutes; use Copilot in Word or Gemini in Docs to draft and improve a report; compare ChatGPT, Claude, Gemini, and Microsoft Copilot on the same writing task; use Grammarly to review your own writing.

Unit II: AI for Research, Note-Taking, Data, and Spreadsheets:

NotebookLM (Google): uploading lecture notes, textbooks, PDFs, and research papers as sources; asking questions grounded in uploaded documents; generating study guides, timelines, briefing documents, and audio overviews (podcast-style summaries) from sources; source-grounded answers that cite the exact passage — eliminating hallucination risk.

AI for Research: Perplexity AI (real-time web search with citations), Gemini Deep Research (multi-step research reports), Elicit and Consensus (academic paper search with evidence summaries); verifying AI-generated citations on Google Scholar; why AI fabricates references and how to detect it.

Prompt Engineering: zero-shot, one-shot, few-shot, chain-of-thought, role-based prompting; iterative refinement; negative prompting; building effective prompts for summarization, comparison, explanation, and data extraction; using ChatGPT custom instructions and Claude Projects for persistent context.

AI for Spreadsheets and Data Analysis: Microsoft Copilot in Excel — VLOOKUP, IF, COUNTIF, pivot tables from natural language; Gemini in Google Sheets — formula generation and chart creation; ChatGPT Advanced Data Analysis and Julius AI — uploading CSV files, computing statistics, generating charts, interpreting patterns — no coding required.

AI Integration: Upload your own lecture notes to NotebookLM and generate a study guide and audio overview; use Perplexity AI and Gemini Deep Research to research a topic; verify citations on Google Scholar; use Wolfram Alpha or Symbolab to solve and verify equations from your current semester subjects; use Copilot in Excel or ChatGPT ADA to analyse a real dataset.

Unit III: AI for Visual Design, Diagrams, and Media:

AI for Image Generation: DALL-E 3 (via ChatGPT), Canva AI (text-to-image, Magic Design, background remover), Adobe Firefly, Ideogram — writing effective image prompts (style, subject, composition, colour); iterative prompt refinement for posters, diagrams, and mockups.

AI for Diagrams and Visual Thinking: Napkin.ai — converting text or concepts into visual diagrams, charts, and infographics automatically; Whimsical AI — AI-generated mind maps, flowcharts, and wireframes from a description; Miro AI — collaborative whiteboard with AI summarization and diagramming.

AI for Audio and Video: ElevenLabs — text-to-speech and voice cloning for presentations and voiceovers; Suno and Udio — AI music generation for project videos; Runway and Pika — generating short video clips from text prompts; HeyGen — AI avatars for video presentations; current limitations and appropriate use cases.

AI Content Detection and Critical Evaluation: GPTZero, Originality.ai for AI-text detection; Google Lens and reverse image search for AI-generated images; fact-checking workflows — cross-referencing primary sources; deepfakes and synthetic media — recognition, risks, and responsible creation.
AI Integration: Use Napkin.ai to convert a paragraph of text into a diagram; use Whimsical AI to generate a mind map of a course topic; use DALL-E 3 or Canva AI to create an event poster through 3 prompt iterations; use ElevenLabs to create a voiceover; test GPTZero on 5 samples and analyse confidence scores.

Unit IV: AI Across Engineering Disciplines:
AI in Mechanical Engineering: generative design in CAD (Autodesk Fusion 360 — AI-optimized structures for weight and load); predictive maintenance using sensor data and ML; AI-assisted FEA simulation; manufacturing process optimization; digital twins.
AI in Civil and Electrical Engineering: structural health monitoring using ML; smart grid load forecasting and energy management; AI for circuit simulation and PCB layout optimization (Altium AI); VLSI design automation — floor planning and routing; power systems fault detection.
AI in Electronics and Communication: AI-assisted signal processing and filtering; speech recognition for embedded systems; computer vision for industrial quality inspection; AI in antenna design and spectrum management; MATLAB AI and Deep Learning Toolbox.
AI in Computer Science and allied branches: GitHub Copilot for code generation, explanation, and review; AI-powered testing (Testim, Selenium AI); AI in cybersecurity — anomaly detection and intrusion detection; no-code/low-code platforms (Bubble, Glide, Webflow AI); AI for data science (Jupyter AI, DataRobot).
AI Integration: Each student identifies 5 real AI tools in their own engineering branch using Claude or ChatGPT and verifies each using official product pages or research publications; present findings in a 3-minute class presentation with a structured comparison table.

Unit V: Responsible AI, Ethics, and Capstone Project:
AI bias and fairness: sources of bias (data, model, deployment); real documented cases — Amazon hiring algorithm (2018), COMPAS recidivism (2016), facial recognition disparities (NIST 2019); fairness metrics; misinformation, deepfakes, and synthetic media risks.
Privacy and intellectual property: what happens to data entered into public AI tools; PII risks; data retention policies of ChatGPT (OpenAI), Claude (Anthropic), and Gemini (Google); who owns AI-generated content; copyright status globally; academic plagiarism with AI; personal data safety protocol.
AI governance frameworks: EU AI Act risk tiers (unacceptable, high, limited, minimal); UNESCO AI Ethics Recommendation; India's NITI Aayog Responsible AI Guidelines; India Digital Personal Data Protection Act 2023 (DPDP Act); environmental cost of AI — energy and carbon footprint of LLM training and inference.
Capstone Mini-Project: each student (or pair) defines a real engineering problem, selects at least 3 AI tool categories from Units I–IV, produces a quality-verified deliverable (report, presentation, diagram, or automation), and presents in 5 minutes. Evaluation: problem clarity, tool selection, output quality, human refinement, responsible AI disclosure.
AI Integration: Read and compare data retention policies of ChatGPT, Claude, and Gemini; classify 5 daily-use AI systems using EU AI Act risk tiers; use GPTZero and Google Lens to detect AI-generated content; design and present the Capstone Mini-Project integrating tools from all five units.

Reference Books:

1. Ethan Mollick, Co-Intelligence: Living and Working with AI, Portfolio/Penguin, 2024.
2. Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson Education, 2020.
3. Reid Blackman, Ethical Machines, Harvard Business Review Press, 2022.
4. Google – Fundamentals of AI – grow.google/intl/en_in/ai-learning-resources
5. Microsoft – AI Skills Navigator – microsoft.com/en-us/ai/ai-skills
6. Andrew Ng – AI for Everyone (Coursera, free to audit) – coursera.org/learn/ai-for-everyone
7. Elements of AI – University of Helsinki (free, beginner) – elementsofai.com
8. AI Assistants: ChatGPT – chat.openai.com | Claude – claude.ai | Gemini – gemini.google.com | Microsoft Copilot – copilot.microsoft.com
9. Research & Notes: NotebookLM – notebooklm.google.com | Perplexity AI – perplexity.ai | Elicit – elicit.org | Consensus – consensus.app
10. Presentations & Docs: Gamma.app | Notion AI – notion.so | Beautiful.ai | Copilot in Office 365 | Gemini in Google Workspace
11. Diagrams & Design: Napkin.ai | Whimsical AI – whimsical.com | Miro AI – miro.com | Canva AI – canva.com | DALL-E 3 via ChatGPT
12. Data & Spreadsheets: ChatGPT Advanced Data Analysis | Julius AI – julius.ai | Copilot in Excel | Gemini in Sheets
13. Math & Equations: Wolfram Alpha – wolframalpha.com | Symbolab – symbolab.com | Mathway – mathway.com | Microsoft Math Solver – math.microsoft.com | Photomath – photomath.com | Desmos – desmos.com | GeoGebra – geogebra.org
14. Audio & Video: ElevenLabs – elevenlabs.io | Suno – suno.com | Runway – runwayml.com | HeyGen – heygen.com
15. Detection & Automation: GPTZero – gptzero.me | Originality.ai | Google Lens | Make.com | GitHub Copilot

Online Resources:

1. https://www.youtube.com/@JoyofLearningCSE/featured
2. EU AI Act – eur-lex.europa.eu | UNESCO AI Ethics – unesco.org | NITI Aayog – niti.gov.in
3. India DPDP Act 2023 – meity.gov.in | ProPublica COMPAS – propublica.org | MIT Technology Review – technologyreview.com |

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26PC05101T	PC	Data Structures & Efficient Problem Solving (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	2	0	2	0	3

Pre-Requisites: Basic knowledge of Python programming, programming fundamentals, problem-solving and algorithmic thinking, object-oriented programming, and elementary mathematics/discrete mathematics.

Course Objectives:

- To introduce fundamental concepts of data structures and their applications using Python.
- To develop the ability to analyse time and space complexity of algorithms.
- To implement linear and non-linear data structures using Python.
- To apply appropriate data structures for efficient problem solving.
- To develop skills in algorithm design, recursion, and optimization techniques.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply basic data structure concepts and analyse algorithm complexity using Python. (L3)

CO2: Implement searching and sorting techniques for efficient data processing. (L3)

CO3: Develop programs using linear data structures — stacks, queues, and linked lists — in Python. (L4)

CO4: Design and implement non-linear data structures including trees and heaps using Python. (L4)

CO5: Apply graph traversal and hashing techniques to solve real-world computational problems. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	2		1				2	3	3
CO2	3	3	2	2	2		1				2	3	3
CO3	3	3	3	2	2		1				2	3	3
CO4	3	3	3	2	2		1				2	3	3
CO5	3	3	3	3	2		1				2	3	3

Unit I: Introduction, Algorithm Analysis, and Linear Structures:

Core Content: Introduction to data structures; Abstract Data Types (ADTs); algorithm analysis — time and space complexity; Big-O, Big-Omega, Big-Theta notations. Lists in Python: operations on lists; static vs. dynamic memory allocation. Searching techniques: Linear Search, Binary Search, Interpolation Search — analysis and comparison. Sorting techniques: Bubble Sort, Selection Sort, Insertion Sort — algorithms, trace, and complexity analysis.

AI Integration: Use VisuAlgo and Algorithm Visualizer to animate sorting and searching; use ChatGPT to explain Big-O complexity with concrete examples; use Python Tutor to trace memory representation of data structures.

Unit II: Stacks and Queues:

Core Content: Stacks: LIFO principle, push, pop, peek; implementation using Python lists; applications — expression evaluation (infix to postfix, postfix evaluation), parenthesis checking. Queues: FIFO principle, enqueue, dequeue; types — Circular Queue, Priority Queue, Deque; implementation using Python lists and linked structures. Applications of queues: BFS simulation, scheduling, real-world analogies.

AI Integration: Use VisuAlgo to animate stack and queue operations; use ChatGPT to generate edge cases (overflow, underflow, circular behaviour); use GitHub Copilot to implement expression evaluation.

Unit III: Linked Lists:

Core Content: Singly Linked List: node structure, insertion at head/tail/position, deletion by value/position, traversal, reversal. Doubly Linked List: forward and backward traversal, insertion, deletion; Circular Linked List: traversal and operations. Applications: polynomial representation and addition, browser history simulation (doubly linked list), symbol table implementation.

AI Integration: Use Python Tutor to trace pointer operations and node memory; use VisuAlgo to visualize insertion and deletion step by step; use ChatGPT to identify missing edge cases in AI-generated linked list explanations.

Unit IV: Trees and Heaps:

Core Content: Trees: terminology, properties; Binary Trees — representation, traversals (in order, preorder, post order, level order). Binary Search Trees (BST): insertion, deletion, search; applications — autocomplete, range queries, dictionary lookup.

Heaps: Min-Heap and Max-Heap, array representation, heapify, Build-Heap; Heap Sort — algorithm and complexity.

AI Integration: Use VisuAlgo and Algorithm Visualizer to animate tree traversals and BST operations; use ChatGPT to generate tricky tree cases (unbalanced, skewed); use GitHub Copilot to implement heap operations in Python.

Unit V: Graphs and Hashing:

Core Content: Graphs: terminology, types (directed, undirected, weighted); representation — adjacency matrix and adjacency list.

Graph traversals: BFS (queue-based) and DFS (stack/recursion-based); applications — shortest path, cycle detection, social networks.

Hashing: hash functions, properties of a good hash function; collision resolution — chaining (separate chaining), open addressing (linear probing, quadratic probing, double hashing); load factor.

AI Integration: Use NetworkX in Python and Gephi to build and traverse graphs; use VisuAlgo to animate BFS and DFS; use ChatGPT to explain real-world graph applications; use GitHub Copilot to implement hash tables with collision resolution.

Text Books:

1. Shriram K. Vasudevan, Abhishek S. Nagarajan & Karthick Nanmaran, Data Structures Using Python, Wiley India.
2. Michael T. Goodrich, Roberto Tamassia & Michael H. Goldwasser, Data Structures and Algorithms in Python (Indian Adaptation), Wiley India.

Reference Books:

1. Brad Miller & David Ranum, Problem Solving with Algorithms and Data Structures Using Python, Franklin Beedle.
2. Mark Allen Weiss, Data Structures and Algorithm Analysis in C, Pearson Education.

Online Resources:

1. NPTEL – Data Structures and Algorithms – nptel.ac.in
2. VisuAlgo – visualgo.net

3. Python Tutor – pythontutor.com
4. NetworkX – networkx.org
5. GeeksforGeeks – geeksforgeeks.org
6. Algorithm Visualizer – algorithm-visualizer.org

Recommended Free & Open-Source Data Structures / AI Tools:

1. **Language & Environment:** Python, Jupyter Notebook – implementation and tracing.
2. **Visualization:** VisuAlgo, Algorithm Visualizer, Python Tutor – animate sorting, traversal and memory operations.
3. **Graph Tools:** NetworkX, Gephi – graph build and traversal.
4. **AI Assistants:** ChatGPT, GitHub Copilot – complexity explanation, edge-case generation and implementation.

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26BSHB106P	BS	**Physics for Advanced Computing Lab** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	0	0	0	3	1.5

Pre-Requisites: Basic Physics and Mathematics (Class XII level);

Course Objectives:

- To introduce the fundamental principles of optics, diffraction, interference, spectroscopy, and related phenomena through laboratory experiments.
- To develop practical skills in studying mechanical systems, oscillations, stretched strings, springs, torsional motion, and coupled oscillators.
- To provide hands-on experience in using lasers, photoelectric effect, ultrasonic waves, acoustic grating, and other experimental techniques for determining physical quantities.
- To develop experimental and analytical skills in accurate measurement, data analysis, graph plotting, error estimation, interpretation of results, and effective communication of experimental findings.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply the principles of optics, diffraction, interference, polarization, and spectroscopy to determine optical parameters such as wavelength, refractive index, dispersive power, thickness, and radius of curvature. (L3)

CO2: Analyze the behavior of mechanical systems such as springs, stretched strings, torsional pendulums, and coupled oscillators to determine mechanical constants and frequencies. (L4)

CO3: Apply experimental methods involving lasers, photoelectric effect, ultrasonic waves, and acoustic phenomena to determine physical quantities such as particle size, Planck's constant, and ultrasonic velocity. (L3)

CO4: Conduct physics experiments systematically by using appropriate instruments, experimental techniques, observation procedures, and data-analysis methods to obtain accurate measurements. (L4)

CO5: Interpret and evaluate experimental observations using graphs, calculations, error analysis, and theoretical relationships, and communicate the experimental results effectively. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	3	2		3	3	3		2	2	1
CO2	3	3	2	3	2		3	3	3		2	2	1
CO3	3	3	3	3	2		3	3	3		2	2	1
CO4	3	3	2	3	2		3	3	3		3	2	2
CO5	3	3	2	3	2		3	3	3		2	2	2

List of Experiments:

1. Determination of the dispersive power of the prism.
2. Determination of the thickness of a thin object by the wedge method
3. Determination of the radius of curvature of the lens by Newton's rings
4. Determination of wavelengths of various colours of mercury spectrum using a diffraction grating in the normal incidence method
5. Determination of wavelength using a diffraction grating
6. Determination of particle size using a diode laser

7. Estimation of Planck's constant using the photoelectric effect
8. Determination of spring constant by Hooke's law
9. Determination of the frequency of the tuning fork-Melde's experiment
10. Verification of the three laws of stretched strings using sonometer
11. Determination of frequency of normal modes of the Coupled Oscillator
12. Study of the Refractive Index of Materials using a hollow prism used in surveying
13. Study of oscillations of a mass under different combinations of springs (Loaded Springs)
14. Determination of ultrasonic velocity in liquid (Acoustic grating)
15. Determination of Rigidity modulus of material of a wire-dynamic method (Torsional pendulum)

AI Extension: Python (NumPy, SciPy, OpenCV, Matplotlib) – FREE on Google Colab for optics and EM simulations.

Web Resources:

1. HyperPhysics – hyperphysics.phy-astr.gsu.edu
2. PhET Simulations – phet.colorado.edu
3. https://iitb.vlabs.co.in/discipline.html?discipline=Physical_Sciences

Recommended AI Tools:

1. Python (NumPy, Pandas, Matplotlib)

TensorFlow / Scikit-learn

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26PC05101P	PC	Data Structures & Efficient Problem Solving Lab (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	0	0	0	2	1

Pre-Requisites: Programming fundamentals in Python.

Course Objectives:

- To introduce fundamental concepts of data structures and algorithm design using Python.
- To develop the ability to analyze and compare algorithm efficiency using time and space complexity measures.
- To enable students to implement linear and non-linear data structures for solving computational problems.
- To build problem-solving skills through appropriate selection and application of data structures and algorithms.
- To encourage effective use of AI-assisted tools for visualization, implementation, debugging, and optimization of data structure solutions.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply data structure concepts and analyse algorithm complexity using asymptotic notations and Python implementations. (L3)

CO2: Implement and evaluate searching and sorting algorithms for efficient data processing. (L3)

CO3: Develop applications using linear data structures including stacks, queues, and linked lists in Python. (L4)

CO4: Design and implement non-linear data structures including trees and heaps for problem solving. (L4)

CO5: Apply graph traversal and hashing techniques to develop efficient computational solutions and analyse their performance. (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	3		3	3	3		2	3	3
CO2	3	3	2	2	3		3	3	3		2	3	3
CO3	3	3	3	2	3		3	3	3		2	3	3
CO4	3	3	3	2	3		3	3	3		2	3	3
CO5	3	3	3	3	3		3	3	3		2	3	3

List of Experiments:

Module I: Introduction, Algorithm Analysis & Linear Structures:

Topics: Introduction to Data Structures; Abstract Data Types (ADTs); algorithm analysis; time and space complexity; Big-O, Big-Omega, Big-Theta; lists in Python; searching; sorting.

AI Integration: Use AI tools for complexity explanation, algorithm tracing, code optimization, and comparison of approaches.

Suggested Tools: Python, Jupyter Notebook, VisuAlgo, Python Tutor, Algorithm Visualizer

Experiments:

1. Complexity Analysis of Python Programs — Implement simple programs and analyse time complexity using Big-O notation; compare multiple implementations using AI-generated complexity explanations.
2. Python List Operations and Memory Behaviour — Implement insertion, deletion, searching,

and slicing; compare static and dynamic memory allocation using Python lists.

3. Searching Algorithms Comparison — Implement Linear Search, Binary Search, and Interpolation Search; compare execution time and number of comparisons.
4. Sorting Algorithms Analysis — Implement Bubble Sort, Selection Sort, and Insertion Sort; visualize execution and compare performance on different input datasets**.**

Module II: Stacks and Queues:

Topics: Stacks; expression evaluation; parenthesis checking; queues; Circular Queue; Priority Queue; Deque; queue applications.

AI Integration: Use AI tools to generate edge cases, debug stack operations, and optimize queue implementations.

Suggested Tools: Python, VisuAlgo, GitHub Copilot, Python Tutor

Experiments:

5. Stack Implementation and Applications — Implement stack operations using Python lists and solve parenthesis-matching problems.
6. Expression Conversion and Evaluation — Implement infix-to-postfix conversion and postfix evaluation.
7. Queue Variants Implementation — Implement Queue, Circular Queue, Priority Queue, and Deque.
8. Queue-based Applications — Simulate BFS traversal and scheduling applications using queue structures.

Module III: Linked Lists:

Topics: Singly Linked List; Doubly Linked List; Circular Linked List; applications.

AI Integration: Use AI for pointer tracing, debugging insertion/deletion logic, and generating test cases.

Suggested Tools: Python Tutor, VisuAlgo, Jupyter Notebook

Experiments:

9. Singly Linked List Operations — Implement insertion, deletion, traversal, and reversal operations.
10. Doubly and Circular Linked Lists — Implement forward and backward traversal, insertion, and deletion.
11. Linked List Applications — Develop polynomial addition using linked lists and simulate browser history navigation.

Module IV: Trees and Heaps:

Topics: Binary Trees; tree traversals; Binary Search Tree (BST); heaps; Heap Sort.

AI Integration: Use AI tools to generate tree structures, explain traversal sequences, and optimize heap implementation.

Suggested Tools: Python, VisuAlgo, Algorithm Visualizer, GitHub Copilot

Experiments:

12. Binary Tree Construction and Traversals — Implement inorder, preorder, postorder, and level-order traversals.
13. Binary Search Tree and Heap Operations — Implement BST insertion, deletion, and search; implement Min Heap, Max Heap, and Heap Sort.

Module V: Graphs and Hashing:

Topics: Graphs; BFS; DFS; hash functions; collision resolution.

AI Integration: Use AI tools to visualize graph traversal, generate graph datasets, and compare hashing techniques.

Suggested Tools: Python, NetworkX, Gephi, VisuAlgo

Experiments:

14. Graph Representation and Traversal — Represent graphs using adjacency matrix and adjacency list; implement BFS and DFS.
15. Hash Table Implementation and Analysis — Implement hash tables using chaining and open-addressing techniques; analyse collision behaviour and load-factor impact.

Text Books:

1. Shriram K. Vasudevan, Abhishek S. Nagarajan & Karthick Nanmaran, Data Structures Using Python, Wiley India.
2. Michael T. Goodrich, Roberto Tamassia & Michael H. Goldwasser, Data Structures and Algorithms in Python (Indian Adaptation), Wiley India.

Reference Books:

1. Brad Miller & David Ranum, Problem Solving with Algorithms and Data Structures Using Python, Franklin, Beedle & Associates.
2. Mark Allen Weiss, Data Structures and Algorithm Analysis in C, Pearson Education.
3. Narasimha Karumanchi, Data Structures and Algorithms Made Easy, CareerMonk Publications.
4. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein, Introduction to Algorithms, MIT Press.
5. Aditya Bhargava, Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People, Manning Publications.

Online Learning Resources:

1. NPTEL – Data Structures and Algorithms – nptel.ac.in
2. VisuAlgo – visualgo.net
3. Python Tutor – pythontutor.com
4. Algorithm Visualizer – algorithm-visualizer.org
5. NetworkX Documentation – networkx.org
6. GeeksforGeeks – Data Structures – geeksforgeeks.org
7. Python Documentation – docs.python.org

Recommended Free & Open-Source Data Structures / AI Tools:

1. Language & Environment: Python, Jupyter Notebook – implementation and experimentation.
2. Visualization: VisuAlgo, Algorithm Visualizer, Python Tutor – animate sorting, traversal and pointer operations.
3. Graph Tools: NetworkX, Gephi – graph construction and traversal.
4. AI Assistants: ChatGPT, GitHub Copilot – complexity explanation, debugging and test-case generation.

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26ES03102P	ES	**Engineering Workshop** (Common to all branches of Engineering)	0	0	0	3	1.5

Pre-Requisites: Basic Computing Elements.

Course Objectives:

- To create awareness on safety precautions required at workplace.
- To master fundamental hand tools and traditional manufacturing craftsmanship.
- To practice on manufacturing of components using workshop trades including fitting, Sheet metal and carpentry, foundry and welding.
- To import basic electrical engineering knowledge for House Wiring Practice.
- To demonstrate fundamentals of foundry and welding.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply appropriate safety precautions and procedures while performing various workshop operations. (L3)

CO2: Analyze the selection, application, and operational capabilities of measuring, marking, and cutting tools used in workshop practices. (L4)

CO3: Evaluate and fabricate precision wood and metal fitting joints according to standard specifications and quality requirements. (L5)

CO4: Analyze wire standards, residential switching arrangements, lighting circuits, and basic electrical appliances to ensure safe and effective operation. (L4)

CO5: Evaluate the suitability and application of tools used in foundry and welding sections for different workshop operations. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	2		2	2	3	3	3		2	2	1
CO2	3	2	2		2	2	3	3	3		1	2	1
CO3	3	2	2		1	2	3	3	3		2	2	1
CO4	3	2	2			2	3	3	3		1	2	1
CO5	3	2	2			2	3	3	3		1	2	1

1. Safety Practices & Precautions: Demonstrate the necessity of Workshop hazards, various Personal Protective Equipment and emergency protocols.

AI Integration: Computer vision for safety compliance.

2. Wood working: Introduction to different types of woods, carpentry tools, timber joints

a) Half – Lap joint b) Mortise and Ten on joint c) Corner Dovetail joint or Bridle joint

3. Fitting: Familiarity with different types of tools used in fitting and perform fitting exercises like a) V-fit b) Dovetail fit c) Semi-circular fit

4. Sheet Metal Working: Familiarity with different types of tools used in sheet metal working, Developments of following sheet metal job from GI sheets.

a) Tapered tray b) Conical funnel c) Elbow pipe d) Brazing

5. Electrical wiring: Introduction to electrical tools, safety precautions, wiring materials.

Familiarity with different types of basic electrical circuits and make the following connections.

a) Parallel and series b) Two-way switch c) God own lighting d) Tube light e) Three phase motor f) Soldering of wires

6. Plumbing: Introduction to plumbing tools, pipe fittings, thread cutting, making a simple PVC pipe joint. Preparation of Pipe joints with coupling for same diameter and with reducer for different diameters.

7. Welding Shop: Demonstration and practice on Arc Welding and Gas welding. Preparation of a simple Lap joint/Butt joint.

8. Foundry Trade: Demonstration and practice on Moulding tools and processes, Preparation of Green Sand Moulds for a simple Patterns.

9. Introduction to 3D Printing: Demonstrate the core concepts of 3D printing, Hardware, software, Digital file and 3D printing materials. Familiarise with 3D Printing Process of complex objects.

10. Computer Components: Disassemble and reassemble a desktop computer to understand the role of each component (CPU, motherboard, RAM, HDD/SSD, GPU, etc.).

Hardware Troubleshooting & Peripheral Installation: Simulate common hardware issues (e.g., RAM failure, CPU overheating) and identify the procedures to resolving them.

Peripheral Installation: Install and configure peripheral devices such as printers, scanners, and external hard drives.

Network Setup and Configuration: Set up a small LAN network with routers, switches, and computers. Configure IP addresses, subnet masks, and basic network services.

11. Disassembly and Assembly of equipment such as Bicycle, Gear Box, Braking System: Familiarise with disassembly and assembly process

AI Integration Module: Smart energy monitoring using IoT current sensors; building machine learning load forecasting models; training anomaly detection systems to automatically flag electrical arc or leakage patterns.

Develop predictive load-monitoring models that detect faulty configurations or phantom energy drains.

Text Books:

1. **Workshop Core Reference**: Workship Technology Vol I & II by S.K. Hajra Choudhury, Media Promoters & Publishers.
2. **Workshop Core Reference**: A Course in Workshop Technology by B.S. Raghuwanshi, Dhanpat Rai & Co. 2015 & 2017
3. **Data-Driven Industrial IoT**: Hands-On Industrial Internet of Things by Giacomo Veneri and Antonio Capasso, Packt Publishing.
4. **Basic Workshop Technology:** Manufacturing Process, Felix W.; Independently Published,2019. Workshop Processes, Practices and Materials; Bruce J. Black, Routledge publishers, 5th Edn. 2015.
5. Wiring Estimating, Costing and Contracting; Soni P.M. & Upadhyay P.A.; Atul Prakashan, 2021-22.

Reference Books:

1. Manufacturing Technology (Foundry, Forming and Welding) by P.N. Rao, McGraw Hill.
2. An Introduction to Predictive Maintenance by R. Keith Mobley, Butterworth-Heinemann

Video Links & Online Lectures:

1. **Classical Shopfloor Operations**: NPTEL Workshop Practices Video Series by IIT Kharagpur
2. **Industrial AI & Predictive Analytics**: Predictive Maintenance Models in Python Tutorial
3. **Smart Sensors & Electronics Fabrication**: MIT 6.131: Power Electronics and Smart Grid Interfaces

B. Tech. – I Year II Semester

Course Code	Category	Name of the Course	L	T	Pr.	P	C
26HMHS103L	HM	NSS / NCC / Scouts & Guides / Community Service (Common to all branches of Engineering)	0	0	0	1	0.5

Pre-Requisites: Basic awareness of civic responsibilities, community service, teamwork, discipline, and general social and environmental issues.

Course Objectives:

- To inculcate the spirit of volunteerism, patriotism and social responsibility among students.
- To enable students to understand the structure, objectives and working of NSS, NCC, Scouts & Guides.
- To develop leadership, communication, teamwork and crisis-management skills through experiential learning.
- To equip students with basic first-aid and disaster-preparedness competencies.
- To promote community engagement and sustainable development values through organised service activities.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply the principles and practices of NSS, NCC, Scouts & Guides, and community service to demonstrate responsible citizenship, civic sense, volunteerism, ethical conduct, and active participation in community-development activities. (L3)

CO2: Apply leadership, communication, teamwork, field-craft, camping, disaster-management, and first-aid skills to effectively respond to community, camp, and emergency situations. (L3)

CO3: Apply community needs-assessment, project-planning, social-awareness, environmental-conservation, digital-literacy, and documentation skills to implement and evaluate community service and service-learning activities. (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong, 2 – Moderate, 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1						3	2	2	2		2		
CO2						3	3	3	2		2		
CO3						3	2	3	2		2		

Unit I: Foundation of NSS, NCC, Scouts & Guides and Community Service:

- Introduction to NSS – history, objectives, badge, motto, pledge
- Introduction to NCC – ranks, uniforms, aims, NCC Act 1948
- Scouts & Guides – Baden-Powell legacy, Bharat Scouts & Guides Act
- Community Service – concept, importance and types
- Citizenship, civic sense and social responsibility
- Volunteerism: global and Indian perspective (NYKS, NSS, Red Cross)
- Legal & ethical framework for community service

Unit II: **Leadership, Skill Development, Camping and Disaster Management:**

- Leadership development – theories and models (situational, transformational)
- Team building, communication & conflict resolution
- Parade, drill & physical training basics

- Map reading, navigation and field craft (NCC)
- Camp organisation – planning, logistics, safety protocols
- Adventure activities: trekking, rock-climbing, swimming (awareness)
- Disaster management – concepts, types, roles of NSS/NCC volunteers
- First Aid – CPR, wound management, fractures, snake bite, burn

Unit III: Community Projects, Social Awareness and Service Documentation:

- Community needs assessment – participatory methodology
- Project planning: Swachh Bharat, Pulse Polio, Blood Donation camps
- Literacy campaigns and awareness drives (PadhnaLikhna Abhiyan)
- Environmental conservation – plantation, solid waste, water conservation
- Gender sensitisation and women empowerment programmes
- Digital literacy and e-governance outreach
- Documentation and reporting of service-learning activities
- Reflection, portfolio preparation and field report writing

Text Books:

1. NSS Training Manual, Ministry of Youth Affairs & Sports, Govt. of India, MYAS Publication, 2022.
2. National Cadet Corps – Training Syllabus & Manual, Directorate General NCC, DG NCC, New Delhi, 2021.

Reference Books:

1. Community Development & Social Service, Dr. S. K. Lal, Regal Publications, 2019.
2. Disaster Management – A Holistic Approach, Satish Modh, Macmillan India, 2020.
3. Volunteer Management, Steve McCurley & Rick Lynch, Energize Publications, 2018.
4. Scouting for Boys, Lord Robert Baden-Powell, Oxford University Press (Centenary Ed.), 2008

Web Resources:

1. NSS Official Portal – https://nss.gov.in
2. NCC Official Site – https://indiancc.mygov.in
3. Bharat Scouts & Guides – https://bsgindia.org
4. UN Volunteers – https://www.unv.org
5. NYKS (Nehru Yuva Kendra) – https://nyks.nic.in
6. National Disaster Management Authority – https://ndma.gov.in
7. Swachh Bharat Mission – https://swachhbharat.mygov.in
8. India Volunteers – https://www.indiavolunteers.org

YouTube Resources:

1. NSS Song & Pledge (Official MYA&S) – https://youtu.be/nss_official_01
2. NCC Drill & Parade Training Basics – https://youtu.be/ncc_drill_02
3. Community Service Project Planning – https://youtu.be/community_svc_03
4. First Aid & CPR Tutorial – Red Cross India – https://youtu.be/firstaid_rci_04
5. Disaster Management Awareness – NDMA – https://youtu.be/ndma_dm_05
6. Baden-Powell and the Scout Movement – https://youtu.be/scouts_bp_06

7. Leadership Skills for Youth – https://youtu.be/leadership_youth_07
8. Swachh Bharat Campaign Activities – https://youtu.be/swachh_sb_08

AI-Integrated Activities:

S. No.	AI Activity	Description / Tool
1.	AI-Assisted Community Needs Survey	Use ChatGPT / Gemini to design survey questionnaires; analyse crowd-sourced responses with AI tools to map priority community issues.
2.	Virtual Camp Planning Simulator	Use Notion AI / Trello AI to simulate camp logistics, assign roles and generate contingency plans for adverse scenarios.
3.	Disaster-Risk FAQ Chatbot	Students build a first-aid guidance chatbot using Dialogflow / Botpress and deploy it for community awareness.
4.	AI Social Media Campaign Design	Use Canva AI / Adobe Firefly to create Swachh Bharat / blood-donation awareness posters; schedule with Buffer AI.
5.	Reflective Portfolio with AI Writing Aid	Draft field reports using Grammarly / ChatGPT; critically compare AI-generated drafts with self-written reflections.

B .Tech. – II Year
I Semester

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26BSHB213T	BS	**Biology for Engineers** (Common to all branches of Engineering)	2	0	0	0	2

Pre-Requisites: Basic Sciences (Intermediate level).

Course Objective:

This course introduces biological concepts through the lens of engineering and artificial intelligence. It emphasizes biomolecules, human systems, bio-inspired designs, and modern bioengineering trends, with practical case studies and AI-enabled problem-solving.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: To apply the fundamentals of biology, biomolecules, and human physiological systems in analyzing and solving engineering-related problems. (L3)

CO2: Apply AI tools and computational techniques for biological data analysis, biomolecule visualization, and healthcare applications. (L3)

CO3: Analyze biomolecules and bio-inspired systems for solving engineering problems and developing sustainable technologies. (L4)

CO4: Design biomimetic and bioengineering solutions using interdisciplinary engineering approaches and AI-enabled tools. (L6)

CO5: Evaluate emerging trends in bioengineering such as 3D bioprinting, regenerative medicine, and AI-based diagnostics for societal applications. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	1	2	1	1	1	2	1				2	2	1
CO2	2	3	2	2	2	2	1				2	2	1
CO3	2	3	2	3	3	2	1				3	2	1
CO4	2	3	2	3	3	2	1				3	1	2
CO5	2	3	3	3	3	2	1				3	2	1

Unit I: Foundations of Biology and AI:

Content: Cell as the basic unit of life,Carbohydrates, proteins, lipids, enzymes, vitamins, hormones.

AI Integration: AI-based visualization of biomolecules (e.g., AlphaFold),Case studies on AI in drug discovery.

Free/Open-Source AI Tools:

Protein folding prediction using Alpha Fold, Use of Deep Chem to simulate enzyme-substrate binding, Compare AI-predicted protein structures with experimental data.

Unit II: Biomolecules in Engineering Applications

Content: Carbohydrates in bioplastics (PHA, PLA),Proteins in food engineering (whey protein, plant analogs)

AI Integration: AI-based visualization of biomolecules (e.g., Alpha Fold),Case studies on AI in drug discovery.

Free/Open-Source AI Tools: PLA bioplastic design using ANSYS, AI-based optimization of protein-rich diets, Design of a cellulose-based water filter using AI simulation, Model biodiesel yield prediction using MATLAB.

Unit III: Human Systems and Bio-Designs:

Content: Brain as CPU (EEG, robotic prosthetics),Eye as camera (optical corrections, lens materials).

AI Integration: Heart as pump (ECG, stents),Kidney as filtration (dialysis systems).

Free/Open-Source AI Tools: AI-based ECG anomaly detection, Robotic arm design inspired by neural signals, Use of AI imaging tools to detect cataracts, Simulate stent design using ANSYS.

Unit IV: Bio-Inspired Materials and Mechanisms:

Content: Photosynthesis → photovoltaic cells. Bird flight → GPS and aircraft navigation.

AI Integration: Lotus leaf → superhydrophobic surfaces.Kingfisher beak → bullet train design.

Free/Open-Source AI Tools:

Model solar energy conversion inspired by photosynthesis, AI simulation of lotus-leaf hydrophobicity, Kingfisher-inspired aerodynamic modelling, Use of AI to design self-cleaning surfaces.

Unit V: Emerging Trends in Bioengineering:

Content: Muscular and skeletal scaffolds, 3D bioprinting of tissues (ear, bone, skin)..

AI Integration: AI in medical imaging (MRI, CT scans),AI in diagnostics and predictive healthcare.

Free/Open-Source AI Tools: AI-based tumor detection in MRI scans, 3D bioprinting case study of bone tissue, Use of AI to classify medical images, To Design a scaffold for bone regeneration using ANSYS.

Text Books:

1. Biology for Engineers, Rajendra Singh C and Rathnakar Rao N, Rajendra Singh C and Rathnakar Rao N Publishing, Bengaluru, 2023.
2. Biology for Engineers, Thyagarajan S., Selvamurugan N., Rajesh M.P., Nazeer R.A., ThilagarajW.,Barathi S., and Jaganthan M.K., Tata McGraw-Hill, New Delhi, 2012.

Reference Books:

1. Human Physiology, Stuart Fox, Krista Rompolski, McGraw-Hill eBook. 16th Edition, 2022
2. Biology for Engineers, Arthur T. Johnson, CRC Press, Taylor and Francis, 2011
3. Biomedical Instrumentation, Leslie Cromwell, Prentice Hall 2011.
4. Biology for Engineers, Sohini Singh and Tanu Allen, Vayu Education of India, New Delhi, 2014.
5. Biomimetics: Nature-Based Innovation, Yoseph Bar-Cohen, 1st edition, 2012, CRC Press.
6. Bio-Inspired Artificial Intelligence: Theories, Methods and Technologies, D. Floreano and C. Mattiussi, MIT Press, 2008.
7. 3D Bioprinting: Fundamentals, Principles and Applications by Ibrahim Ozbolat, Academic Press, 2016.

Online Learning Resources:

1. https://nptel.ac.in/courses/121106008
2. https://freevideolectures.com/course/4877/nptel-biology-engineers-other-non-biologists
3. https://ocw.mit.edu/courses/20-020-introduction-to-biological-engineering-design-spring-2009
4. https://ocw.mit.edu/courses/20-010j-introduction-to-bioengineering-be-010j-spring-2006
5. https://www.coursera.org/courses?query=biology
6. https://onlinecourses.nptel.ac.in/noc19_ge31/preview
7. https://www.classcentral.com/subject/biology
8. https://www.futurelearn.com/courses/biology-basic-concepts

AI Tools for the Course:

1. Deep Chem – drug discovery simulations.
2. Alpha Fold – protein structure prediction.
3. Generative Design AI – biomaterial innovation.
4. ANSYS – biomimetic engineering simulations.
5. MATLAB Simulink – biological system modeling.
6. AI Medical Imaging Tools – diagnostics and healthcare.

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26HMHS204T	HM	Universal Human Values – Understanding Harmony & Ethical Human Conduct (Common to all branches of Engineering)	3	0	0	0	3

Pre-Requisites: Basic understanding of ethics, human values, society and environmental responsibility, along with openness to self-reflection and value-based learning.

Course Objectives:

- To introduce the need for value education and the crisis of values in modern society through self-exploration.
- To develop understanding of harmony within the individual through the Self-Body relationship.
- To build awareness of harmony in the family and society through trust and the comprehensive human goal.
- To Explore harmony in nature and existence through the four orders and their interconnectedness.
- To apply ethical human conduct principles to professional engineering practice and AI ethics.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply the concepts of universal human values, natural acceptance, and the purpose of human life as articulated in Indian value philosophy to personal and professional decision-making. (L3)

CO2: Apply the framework of values-based living to personal relationships, family harmony, and societal well-being in daily life contexts. (L3)

CO3: Analyze ethical dilemmas in engineering practice using the lens of universal human values and differentiate value-driven from interest-driven conduct. (L4)

CO4: Evaluate current engineering and social practices for their alignment with sustainability, ecological balance, and universal human order. (L5)

CO5: Create a personal value charter and an action plan for ethical engineering practice using AI reflection tools and collaborative dialogue. (L6)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1						2	3	1	2		2		
CO2						3	3	2	2		2		
CO3						3	3	2	2		3		
CO4						3	3	2	2		3		
CO5						3	3	3	3		3		

Unit I: Introduction to Value Education and Human Consciousness:

Content: Need for Value Education: Crisis of values in modern society; Human Being: A co-existence of Self (Jeevan) and Body; Consciousness as a distinguishing feature,Natural Acceptance: The innate human capacity to know what is right; Activities of the Self: Knowing, Assuming, Recognising, Fulfiling,Difference between Animal Consciousness and Human Consciousness.

AI Integration: AI Tool: ChatGPT to present ethical dilemmas and compare AI responses with naturally accepted human values. Reflection Activity: Day One or Notion AI journalling assistant for natural acceptance documentation. Discussion: Can AI ever have human values?

Unit II: Harmony in the Individual:

Content: Harmony between Self and Body: Needs of the Self (happiness, trust, etc.) vs. Needs of the Body (food, shelter, clothing); Four Dimensions: Icchha (desire), Vichar (thought), Anubhav (experience), Bodh (realisation),Feelings in Relationships: Nine feelings — Trust, Respect, Affection, Care, Guidance, Reverence, Glory, Gratitude, Love; Understanding Happiness and Prosperity,Self-regulation vs. External Regulation: Inner motivation and its role in ethical conduct.

AI Integration: AI Tool: Replika (free AI companion) for exploring emotional needs conversations and reflecting on AI limitations. Activity: Well-being Wheel using Canva; AI-assisted gap identification between material and value-based happiness.

Unit III: Harmony in the Family and Society:

Content: Family as the Basic Unit of Living: Roles and responsibilities in family; Trust as the Foundational Value in Relationships; From Family to Society: Comprehensive Human Goal, Universal Human Order: Five dimensions of human existence (Individual, Family, Society, Nature, Existence); Social Justice: Ensuring rights, duties, and dignified participation.

AI Integration: AI Tool: ChatGPT to generate family conflict scenarios analysed through trust and respect. Case Study: AI-assisted mapping of historical social movements onto the five-order framework. Project: community service observation documentation.

Unit IV: Harmony in Nature and Existence:

Content: Four Orders of Nature: Material, Plant/Bio, Animal, Human; Interconnectedness: Mutual fulfilment among the four orders; Ecological Balance: Current ecological crises and human responsibility.Sustainable Development Goals (SDGs) and Value-based Engineering; Right Understanding of Existence: Co-existence of units in space.

AI Integration: AI Tool: Global Forest Watch for deforestation data analysis correlated with ecological values. AI Tool: ChatGPT to generate sustainable engineering project ideas critiqued via ecological harmony principles. Activity: campus ecological audit using Google Earth.

Unit V: Ethical Human Conduct and Professional Ethics:

Content: Right Understanding, Right Feeling, Right Action: The value-ethics-skill triad; Professional Ethics: Codes of conduct for engineers (ASME, IEEE, IEI),Corruption, Dishonesty, and Short-term Thinking: Causes and value-based remedies,Vision for a Value-based Society, AI Ethics: Bias, fairness, transparency, and accountability.

AI Integration: AI Tool: IBM AI Fairness 360 to demonstrate AI bias using sample datasets. Case Study: Bhopal Gas Tragedy / Challenger Disaster through professional ethics lens. Capstone: Personal Value Charter drafted with ChatGPT assistance.

Text Books:

1. Gaur, R. R., Sangal, R., &Bagaria, G. P. (2019). A Foundation Course in Human Values and Professional Ethics (3rd ed.). Excel Books.
2. Nagraj, A. (2015). Human Values and Professional Ethics: Universal Human Order. Jeevan Vidya Prakashan.

Reference Books:

1. Subramanian, C. V. (2020). Professional Ethics and Human Values. Oxford University Press.
2. Bajpai, S. C. (2019). Engineering Ethics: Concepts and Cases. McGraw-Hill Education.
3. Flood, R. L. (2018). Rethinking the Fifth Discipline: Learning within the Unknowable. Routledge.
4. Covey, S. R. (2020). The 7 Habits of Highly Effective People (30th Anniversary ed.). Simon & Schuster.
5. Singer, P. (2021). Practical Ethics (4th ed.). Cambridge University Press.

Online Learning Resources:

1. AICTE – Value Education Resources: aicte-india.org
2. UNESCO – Ethics of AI: www.unesco.org/en/artificial-intelligence/recommendation-ethics
3. IEEE Ethics Resources: ethics.iit.edu
4. NPTEL – Human Values & Professional Ethics: swayam.gov.in
5. Jeevan Vidya Andolan: www.jeevanvidya.info

YouTube / Video Resources:

1. YouTube: NPTEL – Human Values & Professional Ethics – www.youtube.com/c/nptel
2. YouTube: TED-Ed – Ethics and Moral Philosophy – www.youtube.com/@TED-Ed
3. YouTube: Michael Sandel – Justice: What's the Right Thing to Do? – search on YouTube
4. YouTube: AI Ethics Overview – IBM Technology – www.youtube.com/@IBMTechnology
5. YouTube: Crash Course – Philosophy – www.youtube.com/@crashcourse

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC31201T	PC	**Fundamentals of Artificial Intelligence** (Common to AI&DS, CSE(AI) & CSE(AI&ML))	2	0	2	0	3

Pre-Requisites: Data Structures; Probability and Statistics; Discrete Mathematics; Python Programming.

Course Objectives:

- To introduce the fundamentals, history, and modern landscape of Artificial Intelligence.
- To develop understanding of intelligent agents and search-based problem solving.
- To explore knowledge representation, logical reasoning, and classical planning.
- To provide foundational knowledge of reinforcement learning and natural language processing.
- To introduce AI applications in robotics, computer vision, and AI ethics and governance.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply AI foundations, agent architectures, environment types, and concepts of the modern AI landscape in analyzing intelligent systems and problem-solving scenarios. (L3)

CO2: Apply uninformed, informed, and adversarial search strategies to solve AI problems. (L3)

CO3: Represent knowledge using propositional and first-order logic and apply planning algorithms. (L3)

CO4: Apply reinforcement learning, NLP models, and transformer-based AI applications. (L3)

CO5: Explain robotics and computer vision principles, evaluate AI ethics and governance frameworks. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1	2	2	1	1				1	3	2
CO2	3	3	2	3	3		1				1	3	3
CO3	3	3	3	3	3	1	1				1	3	2
CO4	3	3	2	3	3	1	1				1	3	3
CO5	3	3	3	3	3	2	1				1	2	3

Unit I: Introduction to AI and Intelligent Agents:

Content: What is AI: definitions, foundations (philosophy, mathematics, neuroscience, linguistics, computer science); history of AI - symbolic AI, expert systems, machine learning era, deep learning, generative AI and LLMs. Intelligent agents: agents and environments; rationality; PEAS framework; environment properties - - observable, deterministic, episodic, static, discrete, single/multi-agent. Agent types: simple reflex, model-based reflex, goal-based, utility-based, learning agents.

AI Integration: Use ChatGPT to apply the PEAS framework to 5 real-world AI systems; use Claude to compare capabilities and limitations of GPT-4, Claude, and Gemini; use NotebookLM to generate study notes from AIMA chapters.

Unit II: Solving Problems By Searching:

Content: Problem-solving agents: state space, initial state, goal test, actions, path cost; toy problems (8-puzzle, 8-queens) and real-world problems (route finding, TSP), Uninformed search:

BFS, uniform-cost search, DFS, depth-limited search, iterative deepening DFS, bidirectional search — completeness, optimality, time and space complexity,Informed search: greedy best-first search; A* search -admissible and consistent heuristics; heuristic design from relaxed problems, Beyond classical search: local search — hill climbing, simulated annealing, genetic algorithms; adversarial search — minimax, alpha-beta pruning; online search agents.
AI Integration: Use ChatGPT to trace BFS, DFS, and A* on a state space graph; use Claude to explain alpha-beta pruning on a game tree with node values; use VisuAlgo or an AI assistant to visualize search algorithm execution.

Unit III: Knowledge Representation and Planning:
Content: Knowledge-based agents: the Wumpus World; propositional logic — syntax, semantics, inference, resolution, model checking (DPLL),First-order logic (FOL): syntax — constants, predicates, functions, quantifiers; unification; forward chaining, backward chaining, resolution; knowledge engineering,Probabilistic reasoning: uncertainty and probability; Bayes' theorem; Bayesian networks — structure, CPTs, variable elimination,Classical planning: STRIPS representation; planning algorithms — forward and backward state-space planning; introduction to MDPs.
AI Integration: Use ChatGPT to trace forward and backward chaining on a knowledge base; use Claude to explain a Bayesian network with a medical diagnosis example; use an AI assistant to explain STRIPS planning with a blocks world example.

Unit IV: Reinforcement Learning and Natural Language Processing:
Content: Reinforcement learning: MDP formulation - states, actions, rewards, policy, value function; passive RL - TD learning; active RL - Q-learning, SARSA; policy search; applications of RL,Deep RL and RLHF: Deep Q-Networks (DQN); policy gradient methods; Reinforcement Learning from Human Feedback (RLHF) - aligning LLMs; applications in games, robotics, and autonomous systems, NLP fundamentals: language models (n-gram); text classification; information retrieval (TF-IDF); information extraction (NER), NLP with transformers: attention mechanism; BERT (bidirectional pre-training) and GPT (autoregressive); fine-tuning and prompt engineering; sentiment analysis, summarization, machine translation.
AI Integration: Use ChatGPT to trace Q-learning on a grid world for 5 steps; use Claude to explain RLHF and how it is used to align LLMs; use an AI assistant to compare BERT and GPT and identify the right model for a given NLP task.

Unit V: Robotics, Computer Vision, and AI Ethics:
Content: Robotics: robot hardware — sensors and actuators; robotic perception, motion planning, SLAM (Simultaneous Localization and Mapping),robotic software architectures — ROS,Computer vision: image formation, image features (edges, corners),image classification,object detection — YOLO, R-CNN; image segmentation. Deep learning for vision: CNN architectures (AlexNet, ResNet),Vision Transformers (ViT),generative AI - GANs and diffusion models,AI ethics and governance: bias and fairness - documented cases (COMPAS, facial recognition disparities), AI safety and alignment; EU AI Act 2024 - risk tiers; India DPDP Act 2023; UNESCO AI Ethics Recommendation; environmental cost of AI.
AI Integration: Use ChatGPT to explain SLAM with a robot navigation scenario; use Claude to explain YOLO object detection; use an AI assistant to classify 5 real AI applications under EU AI Act risk tiers; use NotebookLM to generate an AI ethics study guide.

Text Books:

1. Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson Education, 2020. (Primary — all five units. Chapter previews at aima.cs.berkeley.edu)
2. Nils J. Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann, 1998. (Units I–IV — search, logic, planning, and learning foundations.)

Reference Books:

1. Ethem Alpaydin, Introduction to Machine Learning, 4th Edition, MIT Press, 2020.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press. (Free at deeplearningbook.org)

Online Learning Resources:

1. NPTEL – Introduction to AI, IIT Madras – nptel.ac.in/courses/106106139
2. UC Berkeley CS188 – Intro to AI – inst.eecs.berkeley.edu/~cs188
3. AIMA Python Code – github.com/aimacode/aima-python |HuggingFace NLP Course – huggingface.co/learn/nlp-course
4. EU AI Act – eur-lex.europa.eu | India DPDP Act 2023 – meity.gov.in | UNESCO AI Ethics – unesco.org

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26EC04203T	EC	**Digital Logic and Computer Organization** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	3	0	0	0	3

Pre-Requisites: Knowledge of basic mathematics, number systems, Boolean algebra, logic gates, and fundamental digital electronics concepts.

Course Objectives:

- To understand number systems, binary codes, Boolean algebra, and logic gates; apply K-map minimization and universal NAND/NOR gates to simplify and implement digital logic functions.
- To analyse and design combinational circuits and sequential circuits.
- To understand Von-Neumann architecture, functional units, and bus structures; apply computer arithmetic techniques.
- To understand memory organization covering RAM, ROM, cache mapping, virtual memory, and secondary storage; implement programmable logic using PLA, PAL, and FPGA.
- To understand I/O organization including programmed I/O, interrupt-driven I/O, DMA transfer modes, bus structures, interface circuits, and standard interfaces (USB and PCI).

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply number systems, binary codes, Boolean algebra, and logic gate principles to represent, convert, and simplify digital logic functions using K-maps (L3)

CO2: Design combinational circuits — adders, comparators, multiplexers, and decoders — and analyze sequential circuits including flip-flops, counters, and shift registers. (L4)

CO3: Apply computer arithmetic techniques — 2's complement arithmetic, carry-lookahead addition, Booth's multiplication, and IEEE 754 floating-point operations.(L3)

CO4: Apply concepts of memory organization including semiconductor memories, cache memory, virtual memory, and programmable logic devices (PLA, PAL, FPGA). (L3)

CO5: Describe I/O organization methods — programmed I/O, interrupt-driven I/O, and DMA — and explain bus structures, interface circuits, and standard I/O interfaces (USB, PCI). (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1	1	2		1				1	3	2
CO2	3	3	2	1	2		1				2	3	3
CO3	3	3	2	1	2		1				1	3	3
CO4	3	2	1	1	2		1				2	3	3
CO5	3	2	1	1	2		1				2	2	3

Unit I: Data Representation and Digital Logic Circuits – II:

Content: Binary Numbers, Fixed Point Representation, Floating Point Representation, Number base conversions, Octal and Hexadecimal Numbers, Signed binary numbers, Binary codes,Basic Logic Functions, Logic gates, Universal logic gates, Minimization of Logic expressions, K-Map Simplification, Comparator, Decoders, Multiplexers.

AI Integration: Use an AI/LLM tool to explain binary, octal, decimal, and hexadecimal number systems and walk through conversion steps; solve problems on fixed-point/floating-point representation and complement arithmetic. Use a Python template in Google Colab to verify K-map

simplification against manually derived truth tables. Use an AI/LLM tool to design a 2-to-4 decoder and 4-to-1 multiplexer from NAND gates and trace circuit outputs.

Unit II: Digital Logic Circuits – II and Basic Structure of Computers:

Content: Sequential Circuits, Flip-Flops (SR, JK, D, T), master-slave flip-flops, flip-flop conversions, Binary counters, Registers, Shift Registers (SISO, SIPO, PISO, PIPO), Ripple counters, Synchronous counters,Computer Types, Functional units, Basic operational concepts, Bus structures, Software, Performance, Multiprocessors and multicomputers, Computer Generations, Von-Neumann Architecture.

AI Integration: Use an AI/LLM tool to explain flip-flop truth tables, excitation tables, and conversion methods (e.g. JK to D). Use a Python template in Google Colab to simulate a 4-bit synchronous up-counter and SIPO shift register, verifying state sequences. Use an AI/LLM tool to explain Von-Neumann architecture, fetch-decode-execute cycle, and compare with Harvard architecture.

Unit III: Computer Arithmetic:

Content: Addition and Subtraction of Signed Numbers, Design of Fast Adders (Carry-Lookahead Adder), Multiplication of Positive Numbers, Signed-operand Multiplication (Booth's Algorithm),Fast Multiplication (Wallace Tree), Integer Division (Restoring and Non-Restoring), Floating-Point Numbers and Operations (IEEE 754).

AI Integration: Use an AI/LLM tool to explain 2's complement addition/subtraction with overflow detection and walk through Carry-Lookahead Adder design, comparing delay versus ripple carry adders. Use a Python template in Google Colab to perform Booth's algorithm multiplication and compare with shift-and-add method. Use an AI/LLM tool to explain IEEE 754 floating-point format with a worked addition example.

Unit IV: Memory Organization and Programmable Logic:

Content: Basic Concepts of Memory Organization, Semiconductor RAM Memories (SRAM and DRAM), Read-Only Memories (ROM, PROM, EPROM, EEPROM, Flash), Speed, Size and Cost,Cache Memories (direct-mapped, set-associative, fully associative), Performance Considerations, Virtual Memories, Memory Management Requirements, Secondary Storage; Programmable Logic Array (PLA), Programmable Array Logic (PAL), FPGA.

AI Integration: Use an AI/LLM tool to explain memory hierarchy and cache operation (hit, miss, LRU/FIFO) with a direct-mapped cache example, calculating hit rate manually. Use a Python template to simulate direct-mapped vs 2-way set-associative cache for a memory address trace. Use an AI/LLM tool to implement a Boolean function using PLA and PAL programming tables and compare flexibility.

Unit V: Input/Output Organization and Interfacing:

Content: Introduction to Input/Output organization, Accessing I/O devices, Programmed I/O, Interrupt-driven I/O, Interrupt handling, Vectored interrupts, Processor examples for I/O operations**,**Direct Memory Access (DMA), DMA transfer modes (burst, cycle-steal, transparent), Buses and bus structures, Synchronous and asynchronous buses, Interface circuits, Serial and parallel communication, Standard I/O interfaces, USB, PCI, Memory-mapped and isolated I/O.

AI Integration: Use an AI/LLM tool to compare programmed I/O, interrupt-driven I/O, and DMA in terms of CPU involvement and throughput, and trace the complete interrupt handling sequence. Use a Python template to simulate programmed vs interrupt-driven I/O and compute CPU

utilization for both. Use an AI/LLM tool to explain synchronous/asynchronous bus timing and compare USB/PCI interface standards.

Text Books:

1. Hamacher, C., Vranesic, Z., & Zaky, S. (2023). Computer Organization (6th ed.). McGraw Hill.
2. Mano, M. M. (2018). Digital Design (6th ed.). Pearson Education.

Reference Books:

1. Mano, M. M. (2017). Computer Systems Architecture (3rd ed.). Pearson.
2. Patterson, D. A., & Hennessy, J. L. (2004). Computer Organization and Design. Elsevier.
3. Roth, C. H. (2003). Fundamentals of Logic Design (5th ed.). Thomson.
4. Stallings, W. (2022). Computer Organization and Architecture (11th ed.). Pearson.

Online Learning Resources:

1. NPTEL Computer Organization: https://nptel.ac.in
2. NPTEL Digital Logic Design: https://nptel.ac.in
3. Google Colab for Python: https://colab.research.google.com
4. VisuAlgo – Visualizing Data Structures: https://visualgo.net

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC05202T	PC	**Algorithms Design & Analysis** (Common to CSE, AI&DS, CSE(AI)& CSE(AI&ML))	3	0	0	0	3

Pre-Requisites: Computational Thinking and Problem Solving with Python.

Course Objectives:

- To understand fundamental algorithm design paradigms — divide and conquer, greedy, dynamic programming.
- To analyse algorithm efficiency and correctness using asymptotic notation, recurrence relations, and amortized analysis.
- To design efficient algorithms for complex problems using appropriate paradigms.
- To understand computational complexity, NP-completeness, and approximation strategies.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply asymptotic notation, amortized analysis, and recurrence solving techniques to analyse algorithm complexity. (L3)

CO2: Design and analyse divide-and-conquer algorithms — sorting, selection, and matrix multiplication. (L4)

CO3: Apply greedy algorithms to graph and optimization problems — MST, shortest path, and scheduling. (L3)

CO4: Design dynamic programming solutions for classic problems — LCS, knapsack, edit distance, and shortest paths. (L4)

CO5: Classify problems into P, NP, NP-Complete, NP-Hard; apply backtracking, branch-and-bound, and approximation algorithms. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	3	2		1				1	3	2
CO2	3	3	2	3	2		1				1	3	3
CO3	3	3	3	3	2		1				1	3	3
CO4	3	3	3	3	3		1				1	3	3
CO5	3	3	3	3	3		1				1	3	3

Unit I: Algorithm Analysis And Mathematical Foundations:

Content: Asymptotic notation: O, Ω, Θ, o, ω; rate of growth comparisons; properties of asymptotic notations; amortized analysis — aggregate, accounting, and potential methods, Recurrence solving: substitution method; recursion tree method; Master theorem — all three cases, extensions, and limitations, Lower bounds: decision tree model; lower bounds for sorting (Ω(n log n)) and searching; adversary arguments.

AI Integration: Use Visu Algo and Python Tutor to visualize recursion trees for Merge Sort and Binary Search; use Wolfram Alpha and SymPy to solve recurrence equations; plot and compare O(log n), O(n), O(n log n), $O(n^2)$ growth curves.

Unit II: Divide And Conquer:

Content: Divide and conquer paradigm: strategy, recurrence formulation, subproblem independence. Sorting: Merge Sort (algorithm, analysis, stable), Quick Sort (partitioning schemes, randomized Quick Sort, expected complexity), Heap Sort review, Searching and selection: Binary Search and variants; finding median and order statistics — randomized selection (expected O(n)); deterministic O(n) selection, Advanced applications: Strassen's matrix multiplication — recurrence, complexity, cache performance; convex hull introduction; Master theorem applications for all divide-and-conquer recurrences.

AI Integration: Use VisuAlgo and Algorithm Visualizer to animate Merge Sort, Quick Sort, and Heap Sort; compare Quick Sort pivot strategies (first, last, random median); trace Strassen matrix multiplication and compare with classical $O(n^3)$.

Unit III: Greedy Algorithms:

Content: Greedy method fundamentals: greedy choice property, optimal substructure, control abstraction. Classic problems: activity selection, fractional knapsack, job sequencing with deadlines - greedy proof techniques. Graph algorithms: Minimum Spanning Tree - Prim's algorithm (priority queue implementation, O(E log V)), Kruskal's algorithm (disjoint set union-find); shortest path - Dijkstra's algorithm (non-negative weights), correctness proofs, Advanced: biconnected components, Huffman coding - optimal prefix-free codes, greedy construction, proof of optimality.

AI Integration: Use Gephi and Cytoscape to visualize MST and shortest path algorithms; use VisuAlgo to animate Prim's, Kruskal's, and Dijkstra's; compare multiple scheduling strategies in activity selection.

Unit IV: Dynamic Programming:

Content: DP principles: optimal substructure, overlapping subproblems; memoization (top-down) vs. tabulation (bottom-up); reconstructing solutions from DP tables,Classic DP problems: Longest Common Subsequence (LCS) — table construction and traceback; 0/1 Knapsack — DP table filling; Edit Distance (Levenshtein) — string alignment; Optimal Binary Search Tree,Advanced DP: Floyd-Warshall all-pairs shortest path — $O(V^3)$, negative edge handling, detecting negative cycles; Travelling Salesman Problem (TSP) — Held-Karp DP (exact); Subset Sum.

AI Integration: Use DP Visualizer and Python Tutor to animate LCS table and knapsack DP matrix; compare memoization vs. tabulation on execution time and memory; use ChatGPT to verify step-by-step DP trace for edit distance..

Unit V: Advanced Topics And Computational Complexity:

Content: Backtracking: N-Queens problem, graph coloring, Hamiltonian cycle — pruning and state space trees. Branch and Bound: TSP — bounding functions, LC-search; comparison with backtracking, Computational complexity: problem classes P, NP, NP-Complete, NP-Hard, polynomial reduction, Cook's theorem (SAT is NP-Complete), classic NP-complete problems - Vertex Cover, Clique, 3-SAT, Hamiltonian Path, Approximation algorithms: 2-approximation for Vertex Cover; greedy set cover,TSP approximation (Christofides-Serdyuk), randomized algorithms : Las Vegas vs. Monte Carlo; Randomized QuickSort expected analysis.

AI Integration: Use Graphviz and Gephi to visualize backtracking state space trees; animate N-Queens and graph coloring; classify 10 problems into P/NP/NP-Hard/NP-Complete using Claude; compare exact vs. approximation TSP solutions.

Text Books:

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 4th Edition, MIT Press, 2022.
2. Jon Kleinberg and Eva Tardos, Algorithm Design, Pearson Education, 2013.

Reference Books:

1. Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani, Algorithms, McGraw-Hill, 2008. (Free PDF at algorithms.wtf)
2. Robert Sedgewick and Kevin Wayne, Algorithms, 4th Edition, Addison-Wesley, 2011.

Online Learning Resources:

1. NPTEL – Design and Analysis of Algorithms, IIT Madras – nptel.ac.in
2. MIT Open Course Ware – Introduction to Algorithms (6.006) – ocw.mit.edu
3. Visu Algo – visualgo.net | Algorithm Visualizer – algorithm-visualizer.org
4. Coursera – Algorithms Specialization, Stanford – coursera.org/specializations/algorithms

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC05203T	PC	**Object-Oriented Design & Programming** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	3	0	0	0	3

Pre-Requisites: Basic knowledge of computer fundamentals, programming concepts (variables, data types, operators, control structures, functions), problem-solving techniques, and familiarity with using a programming environment or IDE.

Course Objectives:

- To identify Java language components and understand how they work together in applications.
- To learn the fundamentals of object-oriented programming — defining classes, invoking methods, using class libraries.
- To extend Java classes with inheritance, dynamic binding, and exception handling.
- To design applications with threads and understand Java concurrency mechanisms.
- To apply Java APIs for file handling, GUI programming with JavaFX, and database connectivity.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Analyse problems, design solutions using OOP principles, and implement them in Java. (L4)

CO2: Design and implement classes to model real-world entities with constructors, methods, and access control. (L4)

CO3: Demonstrate inheritance hierarchies, polymorphic behaviour, method overriding, and dynamic dispatch. (L3)

CO4: Apply exception handling to write robust, fault-tolerant Java programs. (L3)

CO5: Perform Java I/O, JDBC database connectivity, and build GUI applications using JavaFX. (L3)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	2	2		1				1	2	3
CO2	3	3	3	2	2		1				1	2	3
CO3	3	3	2	1	2		1				1	2	3
CO4	3	3	2	1	2		1				1	2	3
CO5	3	2	2	1	2		1				1	2	3

Unit I: OOP Concepts, Data Types, Operators, And Control Statements:

Content: OOP principles: encapsulation, inheritance, polymorphism, abstraction; Java program structure — tokens, statements, command-line arguments, escape sequences, comments, programming style. Data types, variables, operators: primitive types, type casting, scope, literals, symbolic constants, static variables, printf(); operators — arithmetic, increment/decrement, ternary, relational, logical, bitwise; precedence and associativity. Control statements: if, if-else, nested if, switch; iteration — while, do-while, for, for-each, nested loops; break, continue.

AI Integration: Use ChatGPT or GitHub Copilot to generate Java programs for control statements; use draw.io to design flowcharts for conditional and loop constructs; use AI to generate test cases for operators and validate precedence via execution.

Unit II: Classes, Objects, and Methods:

Content: Classes and objects: class declaration, modifiers, class members, object creation, assigning objects; access control (public, private, protected, default); constructors, overloaded constructors; nested classes, final class; passing by value and reference; keyword this. Methods: defining methods, overloaded methods, class objects as parameters, access control, recursive methods, nesting of methods, overriding methods; attributes final and static.Java Collections framework: ArrayList, LinkedList, HashMap, HashSet, TreeMap — selection based on use case.

AI Integration: Use StarUML to design UML class diagrams for Student Management and Banking systems; use ChatGPT to generate class structures (constructors, methods) and refine manually; use draw.io for object interaction diagrams.

Unit III: Arrays, Inheritance, and Interfaces:

Content: Arrays: declaration, initialization, storage in memory, element access, operations, sorting (Arrays.sort), searching; 2D and 3D arrays; arrays as vectors.Inheritance: process, types (single, multilevel, hierarchical), super class Object, inhibiting with final, access control, keyword super, constructor chaining, method overriding, dynamic method dispatch; abstract classes.Interfaces: declaration, implementation, multiple interfaces, nested interfaces, interface inheritance; default and static methods in interfaces; functional interfaces; annotations.

AI Integration: Use draw.io to create UML inheritance hierarchy diagrams (Vehicle → Car → ElectricCar); use AI to generate interface-based programs and modify for multiple inheritance; use ChatGPT to compare abstract class vs. interface with concrete examples.

Unit IV: Packages, Exception Handling, and Java I/O:

Content: Packages: defining and importing packages, class path, access control; Java.lang — Object class, Enumeration, Math, Wrapper classes, auto-boxing/unboxing; java.util — Formatter, Random, Time package (Instant, Temporal Adjusters); date/time formatting. Exception handling: hierarchy of exception classes; keywords throws, throw, try, catch, finally; multiple catch clauses; class Throwable; checked vs. unchecked exceptions; try-with-resources. Java I/O: standard I/O streams; byte streams, character streams; Scanner class; reading and writing files using Java NIO (Files class).

AI Integration: Use draw.io to design multi-package system architecture; use ChatGPT to generate programs with exceptions, identify errors, and fix them; use GitHub Copilot for file I/O programs; use AI to explore Java API (Math, Random, Time).

Unit V: Strings, Multithreading, Jdbc, and Javafx:

Content: String handling: CharSequence interface, String class, String Buffer; methods — extraction, comparison, modification, searching. Multithreaded programming: Thread class, creating threads, thread states, priority, synchronization (synchronized keyword), deadlock and race conditions, inter-thread communication (wait, notify, notifyAll); suspending, resuming, stopping threads.JDBC: architecture, MySQL connector setup, establishing connections, Statement vs. Prepared Statement, ResultSet; CRUD operations. JavaFX GUI: Scene Builder, App window structure, displaying text and images, event handling, node layout, mouse events.

AI Integration: Use ChatGPT to generate and debug multithreaded programs; simulate deadlock and resolve; build a JDBC mini-project — connect Java to MySQL and perform CRUD; design a JavaFX login form using Scene Builder.

Text Books:

1. Anitha Seth and B.L. Juneja, JAVA one step ahead, Oxford University Press.
2. Debasis Samanta and Monalisa Sarma, Joy with JAVA — Fundamentals of Object Oriented Programming, Cambridge University Press, 2023.
3. Paul Deitel and Harvey Deitel, Java 9 for Programmers, 4th Edition, Pearson Education.

Reference Books:

1. Herbert Schildt, The Complete Reference Java, 11th Edition, McGraw-Hill.
2. Y. Daniel Liang, Introduction to Java Programming, 7th Edition, Pearson Education.

Online Learning Resources:

1. NPTEL – Programming in Java – nptel.ac.in/courses/106/105/106105191/
2. Oracle Java Documentation – docs.oracle.com/en/java/
3. StarUML – staruml.io| draw.io – draw.io | GitHub Copilot – github.com/features/copilot

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26EC04202P	EC	Digital Logic and Computer Organization Lab (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	0	0	0	2	1

Pre-Requisites: Knowledge of basic mathematics, number systems, Boolean algebra, logic gates, and fundamental digital electronics concepts.

Course Objectives:

- To develop the ability to analyze and design combinational and sequential digital circuits using Boolean algebra, logic gates, flip-flops, counters, and registers.
- To understand and simulate computer arithmetic operations, memory organization, cache mapping techniques, and their impact on system performance.
- To apply concepts of computer organization in modeling memory and I/O systems, and evaluate different data transfer mechanisms through simulation and analysis.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Design and simplify combinational logic circuits using Boolean algebra, K-maps, and universal gates. (L4)

CO2: Implement and verify the behaviour of sequential circuits including flip-flops, counters, and shift registers. (L4)

CO3: Simulate computer arithmetic operations including signed addition, multiplication, and IEEE 754 floating-point representation. (L4)

CO4: Simulate cache memory mapping techniques and compare their performance for given address traces. (L4)

CO5: Simulate I/O transfer methods (programmed, interrupt-driven, DMA) and compare CPU utilization across methods. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1		2		3	3	3		3	3	2
CO2	3	3	2	1	2		3	3	3		3	2	2
CO3	3	3	2	1	2		3	3	3		3	3	2
CO4	2	3	1	2	2		3	3	3		2	2	2
CO5	2	3	1	2	2		3	3	3		2	2	2

List of Experiments:

Unit I: Number Systems, Logic Gates and K-Map Simplification:

1. Number System Conversion and Boolean Expression Verification

Use a Python template in Google Colab to input a decimal number and observe its binary, octal, and hexadecimal equivalents; verify conversions manually using base conversion algorithms. Input a 4-variable Boolean expression and generate its truth table; compare with manually derived truth table.

2. K-Map Simplification and Logic Gate Verification

Simulate a 4-variable K-map for a given Boolean function, observe groupings, and derive the minimized SOP expression; verify by comparing with the manual K-map solution. Implement

the simplified expression using basic logic gates (AND, OR, NOT) on a simulator and verify output for all input combinations.

3. Universal Gate Implementation — Decoder and Multiplexer Design

Design a 2-to-4 decoder and a 4-to-1 multiplexer using only NAND gates. Trace through the circuit for a given input combination and verify the output against the expected truth table; document the universal gate equivalences used (NOT, AND, OR from NAND).

Unit II: Sequential Circuits and Computer Architecture Simulation:

1. Flip-Flop Truth Table Verification and Conversion

Implement SR, JK, D, and T flip-flops on a digital logic simulator; verify their truth tables and excitation tables. Perform a flip-flop conversion exercise (e.g., JK to D) using the excitation table method, and verify the converted flip-flop's behaviour against the target flip-flop's truth table.

2. Synchronous Counter and Shift Register Simulation

Use a Python template in Google Colab to simulate a 4-bit synchronous up-counter and a 4-bit SIPO shift register; apply clock pulses one at a time and observe the state sequence. Verify the counter follows the correct binary sequence and that the shift register correctly loads serial input into parallel output.

3. Von-Neumann Architecture and Fetch-Decode-Execute Cycle Simulation

Trace through a simplified fetch-decode-execute cycle simulation for a sample instruction set, identifying the role of each functional unit (ALU, Control Unit, Memory, I/O). Compare Von-Neumann and Harvard architecture data flow paths and document differences in a summary table.

Unit III: Computer Arithmetic, Memory, and I/O Simulation:

1. Carry-Lookahead Adder Design and Performance Comparison

Design a 4-bit Carry-Lookahead Adder (CLA) by deriving Generate and Propagate expressions and computing all carry signals in parallel. Compare the propagation delay of the CLA implementation against a ripple carry adder for the same 4-bit addition example.

2. Booth's Algorithm Multiplication Simulation

Using a Python template in Google Colab, perform binary multiplication of two 4-bit signed numbers using both the shift-and-add method and Booth's algorithm. Observe the partial products at each step and compare the number of addition steps required by both methods.

3. IEEE 754 Floating-Point Representation and Addition

Convert given decimal numbers to IEEE 754 single-precision floating-point format (sign, exponent, mantissa); perform a floating-point addition example step by step (align exponents, add mantissas, normalize, round). Verify special cases: zero, infinity, and NaN representation.

4. Cache Memory Mapping Simulation

Using a Python template, simulate a direct-mapped cache for a given set of memory addresses; observe which addresses cause hits and which cause misses, and compute the hit rate. Repeat for a 2-way set-associative cache with the same address trace and compare hit rates between the two mapping techniques.

5. Programmed I/O vs Interrupt-Driven I/O Simulation:

Using a Python template, simulate programmed I/O (polling) and interrupt-driven I/O for a simple data transfer scenario. Observe and compute the number of CPU cycles consumed by each method for the same data size, and explain why DMA is preferred for high-speed bulk transfers.

AI Tools Integration (Lab):

Use AI/LLM tools (ChatGPT, Claude) for explaining number conversions, K-map groupings, flip-flop conversions, and IEEE 754 representation step by step. Python (Google Colab) for simulating counters, shift registers, cache mapping, and I/O transfer methods. Logic circuit simulators (Logisim, CircuitVerse) for combinational and sequential circuit design and verification.

Text Books:

1. Hamacher, C., Vranesic, Z., & Zaky, S. (2023). Computer Organization (6th ed.). McGraw Hill.
2. Mano, M. M. (2018). Digital Design (6th ed.). Pearson Education.

Reference Books:

1. Mano, M. M. (2017). Computer Systems Architecture (3rd ed.). Pearson.
2. Patterson, D. A., & Hennessy, J. L. (2004). Computer Organization and Design. Elsevier.

Online Learning Resources:

1. CircuitVerse – Online Digital Logic Simulator: https://circuitverse.org
2. Logisim Evolution – Open Source Logic Simulator
3. NPTEL Digital Logic Design: https://nptel.ac.in
4. Google Colab for Python: https://colab.research.google.com

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC05202P	PC	Algorithms Design & Analysis Lab (Common to CSE, AI&DS, CSE(AI) & CSM(A&ML))	0	0	0	3	1.5

Pre-Requisites: Basic knowledge of programming, data structures (arrays, linked lists, stacks, queues, trees), discrete mathematics, and problem-solving techniques.

Course Objectives:

- To understand and analyze fundamental algorithm design techniques, computational complexity, and mathematical foundations for efficient problem solving.
- To apply divide-and-conquer, greedy, dynamic programming, backtracking, and branch-and-bound techniques to solve real-world computational problems.
- To design, implement, and evaluate efficient algorithms using Python, including randomized, approximation, and graph algorithms.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Implement and analyse sorting and searching algorithms and verify time complexity empirically using Python (L4)

CO2: Implement divide-and-conquer algorithms and compare their performance with brute-force alternatives. (L4)

CO3: Implement greedy algorithms for MST, shortest path, scheduling, and Huffman coding (L4)

CO4: Implement dynamic programming solutions for classic problems and compare memoization vs. tabulation (L4)

CO5: Implement backtracking, branch-and-bound, and approximation algorithms; classify problem complexity experimentally. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	1	3	2		3	3	3		1	3	2
CO2	3	3	2	3	2		3	3	3		1	3	3
CO3	3	3	3	3	3		3	3	3		1	3	3
CO4	3	3	3	3	3		3	3	3		1	3	3
CO5	3	3	3	3	3		3	3	3		1	3	3

List of Experiments:

Unit I: Algorithm Analysis and Mathematical Foundations:

1. Complexity Visualization and Growth Rate Comparison

Implement Linear Search, Binary Search, Bubble Sort, Merge Sort, and Quick Sort in Python. Measure execution time for each on arrays of size n = 1,000 to 1,000,000 using the time module. Plot the measured execution times vs. n alongside the theoretical O(n), O(n log n), $O(n^2)$, and O(log n) curves using Matplotlib. Verify experimentally that the empirical curves match the predicted asymptotic growth. Use VisuAlgo to animate each algorithm side by side.

2. Recurrence Solving and Master Theorem Verification

Implement Merge Sort and Binary Search recursively in Python. Instrument each recursive call to count the total number of recursive calls made for a given input size n. For Merge Sort (T(n) = 2T(n/2) + O(n)) and Binary Search (T(n) = T(n/2) + O(1)), apply the Master theorem to derive

the closed-form solution. Plot the measured call count vs. n and the Master theorem prediction on the same graph. Use SymPy to verify the recurrence solution symbolically.

3. Amortized Analysis — Dynamic Array and Stack

Implement a dynamic array in Python that doubles its capacity on overflow (similar to Python list internals). Insert 10,000 elements and record the actual cost of each insertion (1 for normal, n for resize). Plot the cumulative cost vs. number of insertions. Show using the aggregate method that amortized cost per operation is O(1). Repeat for a stack with multi-pop operations and verify amortized cost using the accounting method.

Unit II: Divide and Conquer:

4. Sorting Algorithm Comparison — Merge Sort, Quick Sort, Heap Sort

Implement Merge Sort, Quick Sort (with three pivot strategies: first element, random, median-of-three), and Heap Sort in Python. Measure execution time on: (a) random arrays, (b) already-sorted arrays, (c) reverse-sorted arrays of size 1,000 to 100,000. Plot a grouped bar chart comparing all strategies across all three input types. Verify that worst-case Quick Sort (sorted input, first pivot) degrades to $O(n^2)$ and that random pivot avoids this. Animate each sort on a 20-element array using VisuAlgo.

5. Randomized Selection and Order Statistics

Implement the Randomized Select algorithm in Python to find the k-th smallest element in an unsorted array in expected O(n) time. Also implement the deterministic O(n) Median-of-Medians algorithm. Compare execution time of both against Python's built-in sort-then-index (O(n log n)) for k = n/2 (median) on arrays of size 10,000 to 500,000. Plot execution time vs. n for all three methods. Verify empirically that Randomized Select runs in expected O(n).

6. Strassen's Matrix Multiplication

Implement classical $O(n^3)$ matrix multiplication and Strassen's O(n^2.807) algorithm in Python. Measure execution time for n × n matrices with n = 2, 4, 8, 16, 32, 64, 128, 256. Plot execution time vs. n on a log-log scale and verify that the slope difference corresponds to the complexity ratio. Identify the crossover point (the n at which Strassen outperforms classical multiplication on your machine). Discuss the cache performance implications.

Unit III: Greedy Algorithms:

7. Activity Selection and Job Sequencing

Implement the Activity Selection problem using a greedy approach (sort by finish time, select non-overlapping activities). Generate 5 test cases with n = 10, 20, 50 activities and verify that the greedy solution is optimal by comparing with exhaustive search for $n \leq 15$. Implement Job Sequencing with Deadlines: given n jobs each with a deadline and profit, find the maximum-profit subset that can be completed before respective deadlines. Plot profit vs. number of jobs selected for varying inputs.

8. Minimum Spanning Tree — Prim's and Kruskal's

Implement Prim's algorithm using a priority queue (heapq) and Kruskal's algorithm using Union-Find (path compression + union by rank) in Python. Apply both to 5 weighted graphs of increasing size (10 to 500 nodes) generated using NetworkX. Verify that both produce the same MST weight. Measure and plot execution time vs. number of edges for both algorithms. Visualize the MST using NetworkX and Matplotlib, coloring MST edges red on the full graph.

9. Dijkstra's Shortest Path and Huffman Coding

Implement Dijkstra's shortest path algorithm using a min-heap on a weighted directed graph. Find shortest paths from a source to all nodes on 3 test graphs (sparse, dense, random) of 100 nodes. Verify correctness against NetworkX's built-in shortest path. Separately, implement Huffman coding: given a string of at least 100 characters, build the Huffman tree, generate codes, encode the string, and compute the compression ratio. Compare the compressed bit length against ASCII fixed-width (8 bits/char) and verify the prefix-free property.

Unit IV: Dynamic Programming:

10. Longest Common Subsequence and Edit Distance

Implement the LCS algorithm in Python using both memoization (top-down with @functools.lru_cache) and tabulation (bottom-up 2D table). Print the DP table and trace back the actual LCS string for 5 string pairs. Implement the Edit Distance (Levenshtein) algorithm for the same pairs. Print the edit distance table and recover the sequence of insert, delete, and substitute operations. Measure and compare execution time and memory usage of memoization vs. tabulation for strings of length 100 to 1,000.

11. 0/1 Knapsack and Subset Sum

Implement the 0/1 Knapsack problem using tabulation and memoization in Python. Given n items with weights and values (n = 10 to 50) and a knapsack capacity W, compute the maximum value and print the selected items by tracing back the DP table. Implement Subset Sum as a special case (value = weight, target = W). For $n \leq 20$, verify the DP result against an exhaustive search (2^n subsets). Plot DP table fill time vs. $n \times W$ to demonstrate pseudo-polynomial complexity.

12. Floyd-Warshall All-Pairs Shortest Path

Implement the Floyd-Warshall algorithm in Python for all-pairs shortest path on a weighted directed graph (with possible negative edges but no negative cycles). Run on 3 test graphs of 10, 50, and 100 nodes. Print the distance matrix after every k-th iteration (k = 1, 5, 10) to visualize how paths are refined. Detect and report negative cycles. Compare execution time with running Dijkstra from each source node (for non-negative graphs). Visualize the shortest path between a selected source-destination pair using NetworkX.

Unit V: Advanced Topics and Computational Complexity:

13. Backtracking — N-Queens and Graph Coloring

Implement the N-Queens problem using backtracking in Python. Solve for N = 4, 6, 8, 10, 12 and count the number of solutions and the number of nodes explored in the state space tree. Plot nodes explored vs. N. Visualize the solution board using Matplotlib. Implement Graph Coloring using backtracking: given a graph and k colors, find a valid coloring (or report none exists). Test on 5 graphs of 10–20 nodes. Visualize the colored graph using NetworkX with node colors reflecting the assigned color.

14. Branch and Bound — TSP

Implement the Branch and Bound algorithm for the Travelling Salesman Problem (TSP) in Python using a lower bound based on reduced cost matrices. Solve for complete graphs of n = 5, 8, 10, 12 cities with random edge weights. Compare the optimal tour cost from Branch and Bound with: (a) nearest-neighbour greedy heuristic, (b) Christofides-style 2-approximation. Record and plot solution quality (tour cost) and computation time vs. n for all three approaches. Document the point at which Branch and Bound becomes infeasible within 60 seconds.

15. NP-Completeness Verification and Approximation Algorithms

Implement the 2-approximation algorithm for Vertex Cover (repeatedly pick an uncovered edge, add both endpoints) and the greedy Set Cover algorithm in Python. Verify 2-approximation: for 10 random graphs of 20–50 nodes, compare the approximate vertex cover size vs. the optimal cover (computed by brute force for small graphs). Plot approximation ratio vs. n. Separately, implement Randomized QuickSort and Las Vegas random selection. Run each 100 times on the same input and plot the distribution of comparisons made, verifying expected O(n log n) and O(n) behaviour respectively.

AI Tools Integration (Lab):

Use Google Colab as the primary lab environment — no local setup required. Use Visu Algo (visualgo.net) to animate sorting, graph traversal, and DP table construction before implementing. Use Python Tutor to trace recursive call stacks for Merge Sort, Quick Sort, and DP memoization. Use NetworkX and Matplotlib for graph construction and visualization of MST, shortest paths, and colored graphs. Use SymPy to verify recurrence solutions and Wolfram Alpha to cross-check asymptotic results. Use ChatGPT or Claude to explain algorithm steps, verify DP table traces, and debug implementation errors — all AI-generated code must be tested and documented before submission.

Text Books:

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 4th Edition, MIT Press, 2022.
2. Jon Kleinberg and Eva Tardos, Algorithm Design, Pearson Education, 2013.

Reference Books:

1. Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani, Algorithms, McGraw-Hill, 2008. (Free PDF at algorithms.wtf)
2. Robert Sedgewick and Kevin Wayne, Algorithms, 4th Edition, Addison-Wesley, 2011.

Online Learning Resources:

1. Google Colab – colab.research.google.com | Python Tutor – pythontutor.com
2. VisuAlgo – visualgo.net | Algorithm Visualizer – algorithm-visualizer.org
3. NetworkX – networkx.org | Matplotlib – matplotlib.org |SymPy – sympy.org
4. NPTEL – Design and Analysis of Algorithms, IIT Madras – nptel.ac.in
5. MIT OpenCourseWare – Introduction to Algorithms (6.006) – ocw.mit.edu

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC05203P	PC	Object-Oriented Design & Programming Lab (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	0	0	0	3	1.5

Pre-Requisites: Basic knowledge of computer programming concepts, problem-solving techniques, data types, control structures, functions, and object-oriented programming fundamentals.

Course Objectives:

- To develop a strong foundation in Java programming and object-oriented concepts and apply them to solve computational problems using classes, inheritance, interfaces, collections, and exception handling.
- To design and develop robust Java applications involving file handling, multithreading, JDBC, database connectivity, and JavaFX-based user interfaces.
- To apply software development, debugging, testing, UML modeling, and AI-assisted programming tools for designing efficient and reliable Java applications.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Write and trace Java programs using data types, operators, control statements, and OOP principles (L4)

CO2: Design and implement classes with constructors, overloaded methods, access control, and Java Collections (L4)

CO3: Implement inheritance hierarchies, interfaces, abstract classes, and dynamic dispatch in Java (L4)

CO4: Design multi-package applications with exception handling, file I/O, and Java standard library APIs (L4)

CO5: Build multithreaded programs with synchronization, connect Java to MySQL using JDBC, and develop a JavaFX GUI application. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1	1	2		3	3	3		1	2	3
CO2	3	3	3	2	2		3	3	3		1	2	3
CO3	3	3	3	2	2		3	3	3		1	2	3
CO4	3	3	2	2	2		3	3	3		1	2	3
CO5	3	2	2	1	2		3	3	3		1	2	3

List of Experiments:

Unit I: OOP Concepts, Data Types, Operators, and Control Statements:

1. Java Basics — Data Types, Operators, and Type Casting

Write Java programs to: (a) declare and print variables of all primitive data types (byte, short, int, long, float, double, char, boolean) with their storage sizes using sizeof equivalent; (b) demonstrate implicit and explicit type casting — int to double, double to int, char to int; (c) demonstrate all operator categories — arithmetic, relational, logical, bitwise (AND, OR, XOR, NOT, left shift, right shift), assignment, ternary; (d) print the operator precedence table and write 5 expressions with mixed operators, predict the result manually, then verify by running the

program. Use GitHub Copilot to generate a test harness for each operator category and validate all edge cases.

2. Control Statements — Decision Making and Loops

Write Java programs to: (a) implement a menu-driven calculator using switch that supports +, −, ×, ÷, %, and power operations; (b) print all prime numbers between 1 and 1,000 using a for loop with break; (c) print the following patterns using nested loops — right-angled triangle, inverted triangle, diamond, Pascal's triangle; (d) implement a number guessing game using a while loop — the program generates a random number (1–100) and gives the user a maximum of 7 attempts with higher/lower hints. Use Python Tutor (Java mode) or draw.io to trace and visualize the control flow of each program before coding.

3. Command-Line Arguments and Static Members

Write Java programs to: (a) accept student name, roll number, and 5 subject marks as command-line arguments, compute total and percentage, and display the grade (O/A+/A/B+/B/C/F) using if-elif-else; (b) implement a class Counter with a static variable count that tracks the total number of Counter objects created across all instances — print the count from both the instance and class level; (c) demonstrate the difference between static and non-static methods by implementing a MathUtils class with static utility methods (GCD, LCM, isPrime, factorial) called without creating an object. Design a UML class diagram for each using StarUML before implementation.

Unit II: Classes, Objects, and Methods:

4. Class Design — Student Management System

Design and implement a Student class with fields: rollNo (int), name (String), marks[] (double array for 5 subjects), and a calculated grade (char). Include: (a) a default constructor, a parameterized constructor, and a copy constructor; (b) overloaded constructors — one accepting only rollNo and name, one accepting all fields; (c) a computeGrade() method using the percentage; (d) overloaded display() methods — one prints summary (name and grade), one prints full details; (e) a static method topStudent(Student[] s) that returns the student with the highest percentage. Draw the UML class diagram in StarUML before coding. Create an array of 5 Student objects and demonstrate all methods.

5. Method Overloading, Recursion, and this Keyword

Write a Java class Calculator with overloaded add() methods that accept: (a) two integers; (b) three integers; (c) two doubles; (d) an integer array (sum of all elements). Demonstrate that Java resolves the correct method at compile time (static dispatch). Separately, implement a RecursionDemo class with recursive methods for: factorial (both recursive and iterative — compare for n = 1 to 20), Fibonacci (recursive — trace the call tree for n = 8 using Python Tutor), binary search on a sorted array, and power(base, exp) using divide and conquer. Demonstrate the this keyword: use this() to call one constructor from another (constructor chaining) and this.field to resolve name shadowing.

6. Java Collections — ArrayList, HashMap, and TreeMap

Write programs demonstrating: (a) ArrayList — store 10 student names, sort alphabetically (Collections.sort), search using indexOf, remove duplicates using a LinkedHashSet, and iterate using for-each and iterator; (b) HashMap — build a word frequency counter that reads a sentence, splits into words, and stores word → count pairs; print the top 5 most frequent words; (c) TreeMap — store student roll numbers and marks in a TreeMap (auto-sorted by key), iterate

in ascending and descending order (descendingKeySet()), and find the student with the highest and lowest marks using firstKey() and lastKey(). Compare ArrayList vs. LinkedList on insertion-at-head time for 100,000 elements.

Unit III: Arrays, Inheritance, and Interfaces:

7. Arrays — 1D, 2D, Sorting, and Searching

Write Java programs for: (a) 1D array operations — insert at position, delete by value, reverse in place, find second largest, rotate left by k positions; (b) 2D array operations — matrix addition, matrix multiplication, transpose, find row with maximum sum; (c) sorting — implement Bubble Sort, Selection Sort, and Insertion Sort; compare their execution time on an array of 10,000 random integers; verify result against Arrays.sort(); (d) searching — implement linear search and binary search (iterative and recursive); for binary search, count the number of comparisons for 20 random queries on a sorted array of 1,000 elements and plot comparisons vs. element position.

8. Inheritance — Vehicle Hierarchy and Runtime Polymorphism

Design and implement a multilevel inheritance hierarchy: Vehicle (base) → MotorVehicle → Car and Truck (hierarchical from MotorVehicle). Vehicle has: make (String), model (String), year (int), fuelType (String); abstract method fuelEfficiency(). Car adds: seatingCapacity (int); Truck adds: payloadCapacity (double). Override fuelEfficiency() differently in Car and Truck. Demonstrate: (a) constructor chaining using super() — print the constructor call order; (b) method overriding — call the same method on a Vehicle reference pointing to Car and Truck objects and verify dynamic dispatch (runtime polymorphism); (c) final method and final class — make a method in MotorVehicle final and verify that Car cannot override it. Draw the full UML hierarchy in StarUML before coding.

9. Interfaces — Payment System and Functional Interfaces

Design a payment processing system using interfaces: (a) Payable interface with processPayment(double amount), getReceipt(), and refund(double amount) methods; implement CreditCardPayment, UPIPayment, and NetBankingPayment — each with different processing logic; demonstrate multiple interface implementation by creating a PremiumPayment class that implements both Payable and Auditable (a second interface with logTransaction() method); (b) demonstrate default methods in interfaces — add a default validateAmount(double amount) method to Payable that checks if amount > 0; show that implementing classes inherit the default without overriding; (c) implement a functional interface MathOperation with a single abstract method operate(int a, int b) — instantiate it using three lambda expressions: add, subtract, multiply.

Unit IV: Packages, Exception Handling, and Java I/O:

10. Packages and Java Standard Library APIs

Create a multi-package application: package com.university.model (Student, Course classes), package com.university.utils (GradeCalculator with static methods), package com.university.main (Main class). Demonstrate access control: show that a private field in Student is not accessible from com.university.main directly. In the same program, demonstrate Java.lang APIs: (a) Math class — compute sin, cos, log, sqrt, pow, and floor for 5 values; (b) Wrapper classes — parse int from String (Integer.parseInt), convert int to binary/octal/hex strings (Integer.toBinaryString), and demonstrate auto-boxing/unboxing in an

ArrayList<Integer>; (c) java.time API — get the current date and time (LocalDateTime.now()), add 30 days (plusDays), find the day of the week, and format using DateTimeFormatter.

11. Exception Handling — Robust Bank Account Application

Implement a BankAccount class that throws custom exceptions: (a) InsufficientFundsException (checked) — thrown when withdrawal exceeds balance; (b) InvalidAmountException (unchecked) — thrown when a negative or zero amount is passed to deposit or withdraw; (c) AccountFrozenException (checked) — thrown when a frozen account is accessed. Implement deposit(), withdraw(), and transfer() methods that throw these exceptions appropriately. Write a BankingApp with a menu-driven interface that: catches each exception with a separate catch block; uses a finally block to log every transaction attempt (successful or failed) to a log file; uses try-with-resources to auto-close the log file writer. Test with at least 10 scenarios covering all three exception types.

12. Java File I/O — Student Record File System

Build a file-based student record system using Java I/O: (a) write student records (roll number, name, 5 subject marks) to a text file using BufferedWriter and read them back using BufferedReader — parse each line and reconstruct Student objects; (b) read a large text file (at least 500 lines) using Java NIO (Files.readAllLines and Files.lines with Stream API) — count total words, find the longest line, and count lines containing a specific keyword; (c) copy a binary file (any image) using FileInputStream and FileOutputStream in chunks of 4KB — measure and report the time taken; (d) demonstrate RandomAccessFile — write 10 student records, then directly seek to and update the 5th record without rewriting the entire file.

Unit V: Strings, Multithreading, JDBC, and JavaFX:

13. String Handling and StringBuffer

Write Java programs demonstrating: (a) String class methods — length, charAt, indexOf, lastIndexOf, substring, replace, toUpperCase, toLowerCase, trim, split, contains, startsWith, endsWith — apply each to at least 3 test strings; (b) String immutability — demonstrate that concatenating 10,000 strings using + operator in a loop is slow; replace with StringBuilder and measure the speedup; explain why String is immutable in Java (security, thread safety, string pool); (c) StringBuffer vs. StringBuilder — implement a palindrome checker, a word reverser, and an anagram checker using StringBuffer; (d) CharSequence interface — write a method that accepts a CharSequence parameter and works correctly when passed a String, StringBuilder, and StringBuffer.

14. Multithreading — Producer-Consumer and Synchronization

Implement the following multithreaded programs: (a) create 5 threads using both Thread class extension and Runnable interface implementation — name each thread and print its name and ID; demonstrate thread priority (set priorities 1–10 and observe scheduling); (b) implement the Producer-Consumer problem using a shared bounded buffer (size = 5) — producer adds items when buffer is not full, consumer removes items when not empty; use synchronized methods with wait() and notifyAll() to coordinate; run 2 producer threads and 3 consumer threads simultaneously; (c) demonstrate a deadlock — two threads each hold one lock and wait for the other; detect it using thread dump (jstack); then resolve it by enforcing a consistent lock ordering; (d) implement a thread-safe bank account with synchronized deposit() and withdraw() methods — run 10 threads performing simultaneous operations and verify the final balance is correct.

15. JDBC and JavaFX — Student Record Management Application

Build a complete Student Record Management application: (a) JDBC backend — create a MySQL database with a students table (roll_no INT PRIMARY KEY, name VARCHAR, dept VARCHAR, cgpa DECIMAL); implement a StudentDAO class with insertStudent(), getAllStudents(), updateCGPA(rollNo, newCGPA), and deleteStudent(rollNo) using PreparedStatement for all queries (no Statement — prevent SQL injection); demonstrate CRUD by inserting 5 students, displaying all, updating 2, and deleting 1; (b) JavaFX frontend — build a GUI with: a TableView displaying all students (fetched from MySQL via JDBC), a form panel with TextField inputs for roll_no, name, dept, cgpa and Add/Update/Delete buttons, and a Search field that filters the TableView in real time; wire all buttons to the DAO methods; (c) integrate exception handling throughout — catch SQLException and show an Alert dialog to the user on any database error.

AI Tools Integration (Lab):

Use GitHub Copilot in IntelliJ IDEA or Eclipse for code generation and intelligent autocomplete — always review, test, and understand the generated code before submission. Use ChatGPT or Claude to explain error messages, debug NullPointerExceptions and thread deadlocks, and verify logic for inheritance hierarchies and JDBC queries. Use StarUML to design UML class and sequence diagrams before each experiment (design-first approach). Use draw.io to create flowcharts and object interaction diagrams for control flow and method call tracing. Use Python Tutor (Java mode) to visualize recursive call stacks and inheritance constructor chaining. Use Scene Builder for drag-and-drop JavaFX UI design in Experiment 15.

Text Books:

1. Anitha Seth and B.L. Juneja, JAVA one step ahead, Oxford University Press.
2. Debasis Samanta and Monalisa Sarma, Joy with JAVA — Fundamentals of Object Oriented Programming, Cambridge University Press, 2023.
3. Paul Deitel and Harvey Deitel, Java 9 for Programmers, 4th Edition, Pearson Education.

Reference Books:

1. Herbert Schildt, The Complete Reference Java, 11th Edition, McGraw-Hill.
2. Y. Daniel Liang, Introduction to Java Programming, 7th Edition, Pearson Education.

Online Learning Resources:

1. Oracle Java Documentation – docs.oracle.com/en/java/
2. NPTEL – Programming in Java – nptel.ac.in/courses/106/105/106105191/
3. StarUML – staruml.io | draw.io – draw.io | Scene Builder – gluonhq.com/products/scene-builderPython Tutor (Java mode) – pythontutor.com | GitHub Copilot – github.com/features/copilot

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26SE31201S	SE	**AI Assisted Application Development** (Common to CSE, AI&DS, CSE(AI) & CSE(A&ML))	1	0	0	2	2

Pre-Requisites: Programming fundamentals, object-oriented programming, and problem-solving concepts**.**

Course Objectives:

- To introduce students to modern web application development using frontend, backend, and database technologies.
- To enable students to utilise AI-assisted development tools for designing, developing, testing, debugging, and improving applications.
- To develop skills to create responsive and interactive web interfaces using HTML, CSS, and JavaScript.
- To build competency in developing full-stack web applications through integration of client-side, server-side, and database components.
- To promote efficient, collaborative, and responsible software development practices using AI-enabled workflows.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Design and develop responsive web interfaces using HTML and CSS with AI-assisted development tools. (L6)

CO2: Develop interactive client-side applications using JavaScript and AI-assisted coding and debugging techniques. (L6))

CO3: Develop server-side applications and integrate databases for web application development. (L6)

CO4: Build, test, and integrate full-stack web applications using AI-assisted development workflows. (L6)

CO5: Deploy and maintain web applications while applying responsible and effective use of AI tools in software development. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	3	1	3	2	3	3	3		1	2	3
CO2	3	3	3	2	3	1	3	3	3		1	2	3
CO3	3	3	3	2	3	1	3	3	3		1	2	3
CO4	3	3	3	3	3	2	3	3	3		2	2	3
CO5	3	2	2	2	3	2	3	3	3		2	3	3

List of Experiments:

Module I: HTML and CSS: Web Interface Design (Weeks 1–3):

1. AI-Assisted Web Page Design using HTML | HTML5, VS Code

Develop a personal portfolio website using HTML5 semantic elements — header, nav, section, article, footer — incorporating text, images, hyperlinks, ordered/unordered lists, tables, and embedded multimedia (video/audio). Use GitHub Copilot or ChatGPT to generate the initial

page structure; critically review and refine the AI output before finalising. Validate the HTML using the W3C Validator; document all changes made to the AI-generated code and explain each modification.

2. Responsive Web Forms using HTML and CSS | HTML5, CSS3

Design a responsive registration and login form with appropriate input controls (text, email, password, radio, checkbox, select, file, submit) and a styled validation interface. Use an AI tool (GitHub Copilot or ChatGPT) to generate form layout and CSS; compare the AI output with a manually designed version and document the differences. Apply media queries to make the form responsive on desktop and mobile viewports (test at 320px, 768px, 1200px widths).

3. CSS Styling and Responsive Layout Development | CSS3

Develop responsive webpages using CSS selectors (class, ID, attribute, pseudo-class, pseudo-element), typography, colour schemes, the Box Model (margin, border, padding, content), Flexbox (flex-direction, justify-content, align-items, flex-wrap), and CSS Grid (grid-template-columns, grid-template-rows, grid-area). Generate an initial layout using AI assistance; document the properties that were accepted, modified, or rejected. Compare the AI-generated layout with a manually coded version and report differences in code quality and responsiveness

Module II: CSS Components and JavaScript Fundamentals (Weeks 4–6):

4. User Interface Component Development | HTML5, CSS3

Create reusable UI components: a responsive navigation bar (hamburger menu for mobile, horizontal links for desktop), a card grid (image, title, description, button), a responsive dashboard (stats panel, sidebar, content area), and a dropdown menu. Use AI tools to generate component CSS and HTML; evaluate each component for UI consistency, accessibility (alt text, ARIA labels, colour contrast), and responsiveness. Document the AI-generated code, the improvements made, and the rationale for each change.

5. JavaScript Programming Fundamentals | JavaScript ES6

Develop interactive web pages demonstrating: variables (let, const, var — scope differences), operators, control flow (if-else, switch, ternary), loops (for, while, for-of, for-in), functions (declaration, expression, arrow function), arrays (map, filter, reduce, find, forEach), and objects (properties, methods, destructuring, spread operator). Use GitHub Copilot to generate starter code for each concept; test each snippet in the browser console and document cases where the AI-generated code contained errors or inefficiencies.

6. DOM Manipulation and Event Handling | JavaScript, Browser Console

Develop an event-driven to-do list application using DOM operations: select elements (getElementById, querySelector, querySelectorAll), create and append elements (createElement, appendChild, insertBefore), modify content (innerHTML, textContent, setAttribute), and handle events (click, keypress, change, submit, mouseover). Implement dynamic content updates — add task, mark complete, delete task, filter by status. Use an AI tool to generate the event listener logic; identify and fix at least 2 bugs in the AI-generated code before submission.

Module III: Client-Side Advanced and Backend Setup (Weeks 7–9):

7. Client-Side Form Validation and Browser Storage | JavaScript

Implement client-side form validation for a registration form: validate name (non-empty, min 3 chars), email (regex pattern), password (min 8 chars, at least one number and special character), confirm password (match check), and phone (10 digits). Display real-time error messages below

each field without page reload. Store validated form data in LocalStorage (persistent) and SessionStorage (session-only); retrieve and display stored data on page load. Use AI to generate the validation regex patterns; test each against 5 valid and 5 invalid inputs and document the results.

8. API Integration and Dynamic Content Generation | JavaScript, REST APIs

Consume two external public APIs (OpenWeatherMap — current weather by city; REST Countries — country information by name) using the Fetch API and async/await. Parse and display JSON responses dynamically on the webpage — weather card (temperature, humidity, icon, description) and country card (flag, capital, population, languages). Handle API errors (network error, invalid city, 404 response) with user-friendly messages. Use ChatGPT to generate the Fetch API code skeleton; test with 5 valid and 3 invalid inputs and verify error handling.

9. Backend Application Setup using Node.js and Express.js | Node.js, Express.js

Set up a Node.js and Express.js backend server: initialise a project (npminit), install Express (npm install express), create a server (app.js) with app.listen() on port 3000, define 5 routes (GET /, GET /about, GET /students, POST /students, DELETE /students/:id) with appropriate HTTP methods and response JSON. Implement middleware for request logging (log method, URL, timestamp for every request) and CORS. Use GitHub Copilot to generate the Express boilerplate; test all routes using Postman; document the AI-generated code and all manual additions.

Module IV: REST API, Database, and Full-Stack Integration (Weeks 10–12):

10. REST API Development and CRUD Operations | Express.js, Postman

Develop a fully functional RESTful API for a Student resource with all four CRUD operations: POST /students (create — validate required fields), GET /students (read all — support query param filtering by department), GET /students/:id (read one — return 404 if not found), PUT /students/:id (update — partial update supported), DELETE /students/:id (delete — return 204 on success). Use proper HTTP status codes for all responses (200, 201, 204, 400, 404, 500). Use GitHub Copilot to generate route handlers; test all endpoints in Postman with 5 test cases per route; export and submit the Postman collection.

11. Database Integration and Authentication | MongoDB, Node.js, Express.js

Connect the Express.js backend to MongoDB Atlas using Mongoose: define a Student schema (name, roll_no, email, department, marks — with type, required, and validation rules); implement the CRUD API from Week 10 using Mongoose model methods (save, find, findById, findByIdAndUpdate, findByIdAndDelete). Implement user authentication: register endpoint (hash password using bcrypt, store in MongoDB), login endpoint (verify password, generate JWT token using jsonwebtoken), and a protected /profile route (verify JWT using middleware). Use AI to generate the Mongoose schema and bcrypt/JWT code; test with 5 user registrations and login scenarios.

12. Frontend–Backend Integration | React.js, REST APIs

Create a React.js frontend (using Create React App or Vite) with 4 components: StudentList (fetch and display all students from GET /students), StudentForm (add a new student via POST /students with controlled form inputs), StudentCard (display individual student data with edit and delete buttons), and a Navbar. Connect to the Express.js + MongoDB backend from Week 11 using Axios (axios.get, axios.post, axios.put, axios.delete). Handle loading states and error

states in each component. Use GitHub Copilot to scaffold components; identify and fix at least 2 integration issues encountered during testing.

Module V: Testing, Deployment, and Mini Project (Weeks 13–15):

13. Full-Stack Application Testing and Optimization | React.js, Node.js, Jest, Lighthouse

Write unit tests for 3 backend API routes using Jest and Supertest (test POST /students with valid data, invalid data, and duplicate entry). Write 2 React component tests using React Testing Library (render StudentList and verify it displays fetched students; render StudentForm and simulate a form submission). Run Google Lighthouse on the frontend (target: Performance > 80, Accessibility > 90, Best Practices >90); document issues found and apply 3 optimisations (image compression, lazy loading, unused CSS removal). Use ChatGPT to generate test case descriptions; write the actual test code yourself.

14. Deployment and Application Monitoring | Render / Vercel / Railway, GitHub

Deploy the full-stack application: push the codebase to a GitHub repository with a clear README (project description, setup instructions, API documentation, screenshots). Deploy the React frontend to Vercel (connect GitHub repo, set build command and output directory). Deploy the Node.js + Express.js backend to Render or Railway (set environment variables for MongoDB Atlas URI and JWT secret). Test the deployed application end-to-end: register a user, login, add 3 students, update 1, delete 1. Use AI tools to generate the README and troubleshoot any deployment errors; document each error encountered and how it was resolved.

15. AI-Assisted Web Application Mini Project | MERN Stack (MongoDB, Express.js, React.js, Node.js)

Design and develop a complete web application (choose from: Student Management System, Smart Attendance System, Event Registration Portal, Expense Tracker, or Task Management Dashboard) integrating all skills from Weeks 1–14. The project must include: (a) a React.js frontend with minimum 5 components and responsive CSS; (b) a Node.js + Express.js REST API with minimum 6 routes covering full CRUD; (c) MongoDB with Mongoose for data persistence and at least 2 schemas; (d) JWT-based authentication with protected routes; (e) deployment on a public URL with GitHub repository; (f) AI usage documentation — a written report listing every AI tool used, the prompts given, the output received, what was accepted, modified, and rejected. Present the working application in a 10-minute demo.

AI Tools Integration (Lab):

Primary coding tools: GitHub Copilot (VS Code extension — autocomplete and inline suggestions), ChatGPT / Claude (code generation, debugging, explanation), Microsoft Copilot (integrated in Edge browser and Office). For UI design: Figma AI (layout suggestions), v0 by Vercel (React component generation from descriptions). For testing: ChatGPT (generating test case scenarios). For deployment: GitHub Copilot CLI (command-line assistance). Responsible AI use policy: all AI-generated code must be reviewed, tested, and understood before submission; a written AI usage log (tool used, prompt, output, what was changed) must accompany every session submission. Copying AI output without testing or understanding is not acceptable.

Text Books:

1. Jon Duckett, HTML and CSS: Design and Build Websites, Wiley Publications.
2. David Flanagan, JavaScript: The Definitive Guide, 7th Edition, O'Reilly Media.
3. Vasan Subramanian, Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React and Node, Apress.

Reference Books:

1. Ethan Brown, Web Development with Node and Express, 2nd Edition, O'Reilly Media.
2. Robert W. Sebesta, Programming the World Wide Web, 8th Edition, Pearson Education.
3. Andrew Ng, Generative AI for Everyone, DeepLearning.AI.

Online Learning Resources:

1. MDN Web Docs (HTML, CSS, JS) – developer.mozilla.org | W3Schools – w3schools.com
2. React Documentation – react.dev | Node.js Docs – nodejs.org/docs | Express.js – expressjs.com
3. MongoDB Docs – mongodb.com/docs | Mongoose – mongoosejs.com
4. GitHub Skills – skills.github.com | VS Code Docs – code.visualstudio.com/docs
5. FreeCodeCamp – freecodecamp.org | OpenAI Developer Resources – platform.openai.com/docs

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26HMHS206A	MC	Technical Paper Writing and IPR (Common to all branches of Engineering)	2	0	0	0	0

Pre-Requisites: Communicative English.

Course Objectives:

- To distinguish technical communication from general communication and identify the types of technical documents used in engineering.
- To develop competency in structuring formal technical reports including abstracts, literature reviews, and data presentation.
- To apply scientific writing conventions for research proposals and IMRAD-structured papers.
- To understand the fundamentals of Intellectual Property Rights including patents, trademarks, and copyrights.
- To draft patent applications and develop IP strategy awareness using AI-assisted writing and patent search tools.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Recall and understand the principles, types, and structure of technical reports, and explain the fundamentals of Intellectual Property Rights. (L2)

CO2: Apply the principles of technical writing to produce well-structured laboratory reports, project reports, and technical proposals. (L3)

CO3: Analyse technical documents for clarity, coherence, accuracy, and compliance with formatting standards using AI-powered review tools. (L4)

CO4: Evaluate patent documents, prior art, and IP claims to assess novelty and inventiveness of an engineering innovation. (L5)

CO5: Create a complete technical report and draft a patent disclosure for an engineering project, integrating AI tools for writing assistance. (L6)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	1	1	1	1			2	1	3	2	2	1	1
CO2	2	2	2	2			2	2	3	2	3	1	2
CO3	2	3	2	2			2	2	3	2	3	2	2
CO4	2	3	2	3			3	2	2	2	3	1	2
CO5	2	2	3	2			2	3	3	3	3	2	2

Unit I: Foundations of Technical Communication:

Content: Technical Communication vs. General Communication: Precision, objectivity, and audience-awareness; Types of Technical Documents: Reports, Proposals, Manuals, Specifications, Abstracts.Writing Process for Technical Documents: Planning, Drafting, Revising, Proofreading; Audience Analysis: Primary and secondary audiences. Language in Technical Writing: Active vs. Passive voice , Nominalisations, Clarity and conciseness.

AI Integration: AI Tool: Grammarly (free) to analyse a technical paragraph for passive voice overuse. AI Tool: Hemingway Editor to improve readability to Grade 10 or below. Activity: compare before/after AI-edited document versions.

Unit II: Technical Report Writing:

Content: Structure of a Formal Report: Title page, Abstract, Table of Contents, Introduction, Methodology, Results, Discussion, Conclusion, References, Appendices.Abstract Writing: Informative vs. Descriptive abstracts; Literature Review: Searching, evaluating, synthesising sources.Data Presentation: Tables, Figures, Charts; Citation and Referencing: IEEE, APA, and MLA styles; avoiding plagiarism.

AI Integration: AI Tool: Elicit.org for AI-powered literature search. AI Tool: Zotero for citation management and bibliography generation. AI Tool: ChatGPT for first-draft abstract generation. Plagiarism check using Turnitin or Copyleaks..

Unit III: Research Proposals and Scientific Writing:

Content: Research Proposal Structure: Background, Problem Statement, Objectives, Methodology, Timeline, Budget; Scientific Paper Format: IMRAD Structure.Writing Results vs. Discussion: Factual reporting vs. Interpretation; Peer Review Process: blind review, revision, resubmission; Conference Papers vs. Journal Articles.

AI Integration : AI Tool: Semantic Scholar for open-access paper search and IMRAD structure analysis. AI Tool: Writefull for AI feedback on scientific language. Activity: write a research proposal with AI structural review..

Unit IV: Intellectual Property Rights Fundamentals:

Content: Introduction to IPR: Definition, importance, and types — Patents, Trademarks, Copyrights, Trade Secrets, Geographical Indications.Patent System in India: Indian Patent Act 1970, Patent Office, application process; Patentability Criteria: Novelty, Inventive Step, Industrial Applicability.Patent Search: Prior art search using databases; Copyright in Engineering: Software copyright, Open-source licenses (GPL, MIT, Apache).

AI Integration: AI Tool: Google Patents for prior art search. AI Tool: Lens.org for patent document search and analysis. Activity: identify and analyse patent claims in a chosen engineering domain.

Unit V: Patent Drafting and IP Strategy:

Content: Structure of a Patent Application: Title, Abstract, Background, Summary, Detailed Description, Claims, Drawings; Types of Claims: Independent and Dependent claims.Filing a Provisional Patent Application (PPA) in India; IP Strategy for Startups and Engineers: trade secrets vs. patents, licensing, technology transfer.Case Studies: Famous patent disputes (Apple vs. Samsung, Monsanto seed patents, pharmaceutical patents).

AI Integration: AI Tool: ChatGPT to draft patent claims for a mechanical invention and critique for clarity.

AI Tool: IP India Online Portal navigation. Capstone: write a complete patent disclosure document.

Text Books:

1. Markel, M., &Selber, S. (2022). Technical Communication (13th ed.). Bedford/St. Martin's.
2. Ahuja, B. N., & Ahuja, S. S. (2021). Intellectual Property Rights: Patents, Trade Marks, Copyrights and Related Topics (4th ed.). Khanna Publishers.

Reference Books:

1. Lannon, J., & Gurak, L. (2020). Technical Communication (15th ed.). Pearson Education.
2. Anderson, P. V. (2019). Technical Communication: A Reader-Centred Approach (9th ed.). Cengage.

3. Meganathan, R. (2019). Intellectual Property Rights for Engineers. Universities Press (India).
4. Cornish, W. R., & Llewelyn, D. (2019). Intellectual Property: Patents, Copyright, Trademarks and Allied Rights (9th ed.). Sweet & Maxwell.
5. World Intellectual Property Organization (WIPO). (2022). WIPO Intellectual Property Handbook. WIPO Geneva.

Online Learning Resources:

1. IP India – Patent Office: www.ipindia.gov.in
2. Google Patents: patents.google.com
3. Lens.org – Free Patent and Scholar Search: www.lens.org
4. WIPO – Patent Resources: www.wipo.int
5. Purdue OWL – Technical Writing: owl.purdue.edu

YouTube / Video Resources

1. YouTube: NPTEL – Technical English for Engineers – www.youtube.com/c/nptel
2. YouTube: IP India – Patent Filing Guide – search on YouTube
3. YouTube: WIPO – Intellectual Property for Engineers – www.youtube.com/@WIPOChannel
4. YouTube: How to Write a Patent Application – PatentFile – search on YouTube
5. YouTube: Scholarly Writing Tips – Academic English – search on YouTube

B .Tech. – II Year
II Semester

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26BSHB212T	BS	**Probability and Statistics** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	2	0	2	0	3

Pre-Requisites: Linear Algebra and Calculus, Differential Equations and Vector Calculus.

Course Objectives:

- To establish a firm foundation in probability theory, random variables, and their properties to model and quantify uncertainty in engineering contexts.
- To analyse standard probability distributions and apply them to compute probabilities and expected values for real-world discrete and continuous phenomena.
- To formulate and evaluate statistical inferences — including estimation, hypothesis testing, and correlation — to support data-driven engineering decisions.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Use statistical measures such as mean, median, mode, variance, standard deviation, skewness, and kurtosis to analyze and characterize data distributions. (L3)

CO2: Apply the concepts axioms and theorems of probability and the properties of discrete and continuous random variables including expectation and variance. (L3)

CO3: Apply binomial, Poisson, normal, and uniform distributions to compute probabilities and derive distribution parameters for engineering problems. (L3)

CO4: Analyse sampling distributions, central limit theorem implications, and construct confidence intervals for population parameters. (L4)

CO5: Evaluate statistical hypotheses using large and small sample tests and interpret correlation and regression relationships in data. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	1					1				1	2	2
CO2	3	2		1			1				1	2	2
CO3	3	2	1	1	1		1				1	2	2
CO4	3	3	1	2	1		1				1	2	2
CO5	3	3	2	3	1		1				1	2	2

Unit I: Descriptive Statistics:

Content: Introduction to statistics; population vs. sample; primary and secondary data; types of variables: categorical and continuous. Measures of central tendency: mean, median, mode; measures of variability: range, variance, standard deviation; skewness; kurtosis.

AI Integration: Python-based computation of mean, median, mode, variance, and standard deviation using NumPy and Pandas; visualization of data distributions with histograms and box plots using Matplotlib/Seaborn, AI-assisted interpretation of skewness and kurtosis in real datasets. Free/Open-Source AI Tools: Python (NumPy, Pandas, Matplotlib, Seaborn) on Google Colab; SciPy stats module.

Unit II: Probability and Random Variables :

Content: Sample space and events; axioms of probability; addition and multiplication theorems; conditional probability; Bayes' theorem. Random variables: discrete and continuous; probability mass function; probability density function; cumulative distribution function; mathematical expectation and variance.

AI Integration: Python simulation of probability experiments using NumPy random module; visualization of PMF, PDF, and CDF using Matplotlib; AI-assisted derivation and verification of Bayes' theorem problems; simulation of conditional probability scenarios using SciPy. Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib) on Google Colab; SymPy; WolframAlpha free tier.

Unit III: Probability Distributions:

Content: Binomial distribution: properties, mean and variance; Poisson distribution: properties, Poisson approximation to binomial.Normal distribution: properties, standard normal curve, areas under normal curve; uniform distribution.

AI Integration: Python-based generation and plotting of binomial, Poisson, normal, and uniform distributions using SciPy and Matplotlib; AI-assisted parameter estimation and verification; simulation of real-world problems using standard distributions. Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib) on Google Colab; SymPy; Scilab.

Unit IV: Sampling Theory and Estimation:

Content: Population and samples; sampling distributions; central limit theorem: statement and significance; t-distribution; chi-square distribution; F-distribution.Point estimation: properties of estimators; interval estimation: confidence intervals for mean and proportion.

AI Integration: Python simulation of sampling distributions and central limit theorem convergence using NumPy; plotting t-, chi-square, and F-distributions with SciPy; AI-assisted construction and interpretation of confidence intervals. Free/Open-Source AI Tools: Python (NumPy, SciPy, Matplotlib, Pandas) on Google Colab; SymPy; GNU Octave.

Unit V: Testing of Hypothesis and Correlation:

Content: Statistical hypothesis: null and alternative hypothesis; level of significance; large sample tests: z-test for means and proportions; small sample tests: t-test (one sample, two samples, paired); F-test. Chi-square test: goodness of fit, independence; correlation coefficient; rank correlation; regression lines and regression coefficients.

AI Integration: Python-based hypothesis testing using SciPy (z-test, t-test, F-test, chi-square test); AI-assisted interpretation of p-values and test statistics; computation and visualization of correlation matrices using Pandas and Seaborn; regression analysis using scikit-learn and Matplotlib. Free/Open-Source AI Tools: Python (NumPy, SciPy, Pandas, Matplotlib, Seaborn, scikit-learn); statsmodels library.

Text Books:

1. Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2016). Probability & Statistics for Engineers and Scientists (9th ed.). Pearson.
2. Miller, I., & Freund, J. E. (2014). Probability and Statistics for Engineers (8th ed.). Pearson.

Reference Books:

1. Jain, R. K., & Iyengar, S. R. K. (2021). Advanced Engineering Mathematics (5th ed.). Alpha Science International.
2. Ross, S. M. (2014). Introduction to Probability and Statistics for Engineers and Scientists (5th ed.). Academic Press.
3. Gubner, J. A. (2006). Probability and Random Processes for Electrical and Computer Engineers. Cambridge University Press.
4. Johnson, R. A., & Bhattacharyya, G. K. (2014). Statistics: Principles and Methods (7th ed.). Wiley.

Online Learning Resources:

1. NPTEL: Probability and Statistics -- https://nptel.ac.in
2. MIT OpenCourseWare -- Introduction to Probability and Statistics: https://ocw.mit.edu
3. Python SciPy Statistics Documentation: https://docs.scipy.org/doc/scipy/reference/stats.html
4. Google Colab for Python: https://colab.research.google.com
5. NumPy Documentation: https://numpy.org/doc
6. Statsmodels Documentation: https://www.statsmodels.org
7. WolframAlpha (free tier) for verification: https://www.wolframalpha.com

Recommended Free & Open-Source AI/Computational Tools :

1. Python (NumPy, SciPy, Matplotlib, Pandas) -- FREE on Google Colab for statistical computation, distribution analysis, and visualization.
2. SymPy (free, open-source) -- For symbolic probability derivations, distribution functions, and algebraic verification.
3. Seaborn (free, open-source) -- For statistical data visualization including correlation heatmaps, distribution plots, and regression plots.
4. scikit-learn (free, open-source) -- For regression modelling, correlation analysis, and exploratory data analysis.
5. Jupyter Notebook (free, open-source) -- For interactive statistical computation and lab documentation.
6. statsmodels (free, open-source) -- For advanced hypothesis testing, regression analysis, and time series statistics.
7. GNU Octave (free, open-source) -- MATLAB-like numerical computing for statistical and probability workflows.

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26HMHS205T	HM	**Managerial Economics and Financial Analysis** (Common to all branches of Engineering)	2	0	0	0	2

Pre-Requisites: Basic mathematics and engineering economics concepts.

Course Objectives:

- To understand the fundamental concepts of managerial economics, demand analysis, and elasticity to make data-driven business decisions.
- To apply production and cost analysis techniques, including break-even analysis, to optimize organizational resource allocation.
- To analyze various forms of business organizations, market structures, and pricing strategies for competitive business environments.
- To evaluate capital budgeting proposals using NPV, IRR, ARR, and Payback methods for strategic investment decision-making.
- To prepare and interpret financial statements, compute key financial ratios, and employ AI/ML techniques for automated financial analysis.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply fundamental economic concepts such as demand, supply, market structures, and time value of money to analyse basic economic situations. (L3)

CO2: Apply economic theories of demand forecasting, cost analysis, and production functions to engineering project decision-making scenarios. (L3)

CO3: Analyse financial statements, cost-volume-profit relationships, and capital budgeting decisions using traditional and AI-powered financial tools. (L4)

CO4: Evaluate investment alternatives using NPV, IRR, and Payback Period methods and assess business viability under uncertainty. (L5)

CO5: Create a comprehensive mini-project business plan incorporating market analysis, cost estimation, and AI-assisted forecasting. (L6)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1							1	1	2	3	1		
CO2							1	1	2	3	1		
CO3							1	2	2	3	1		
CO4							1	2	2	3	1		
CO5							1	3	3	3	1		

Unit I: Introduction to Managerial Economics:

Content: Managerial Economics: Meaning, Nature, Scope and significance for Engineers; Economic Analysis — Micro vs. Macroeconomics; positive vs. normative economics.Demand Analysis: Concept, Determinants, Demand Function, Law of Demand; Elasticity of Demand: Types, Measurement, Business Applications; Demand Forecasting: Meaning, Need, Methods.

AI Integration: AI Tool: ChatGPT to generate demand scenarios for an engineering product and interpret price elasticity. AI Tool: Google Sheets + FORECAST function for Moving Average and Exponential Smoothing. Activity: AI-assisted regression demand forecasting in Google Colab.

Unit II: Production and Cost Analysis:

Content: Production Concepts and Modern Manufacturing Systems; Production Function: Isoquants and Isocosts; Least-Cost Combination of Inputs.Cost Concepts: Fixed, Variable, Total, Average, Marginal Costs; Break-Even Analysis: Concept, Assumptions, Applications; Margin of Safety and Profit Planning.

AI Integration: AI Tool: Google Sheets/LibreOffice Calc for BEA spreadsheet with dynamic charts. AI Tool: ChatGPT to generate cost functions and identify optimal production level. Activity: AI-assisted CVP analysis using Plotly for cost curve visualisation.

Unit III: Business Organisations and Market Structures:

Content: Forms of Business Organisations: Sole Proprietary, Partnership, Joint Stock Companies, Startups and Digital Enterprises; Public vs Private Sector in Digital Economy. Market Structures: Classification with real-world examples; Pricing Methods and Strategies.

AI Integration: AI Tool: ChatGPT to simulate a duopoly pricing game and find Nash Equilibrium. Activity: analyse pricing strategies of competing tech products. Discussion: AI-driven dynamic pricing in e-commerce (Amazon, Uber Surge Pricing).

Unit IV: Capital Budgeting:

Content: Capital: Introduction, Meaning, Nature and Significance; Capital Budgeting: Concept, Nature, Significance in Modern Enterprises, Role in Strategic Decision-Making.Methods: Payback Method, Accounting Rate of Return (ARR), Net Present Value (NPV), Internal Rate of Return (IRR), Profitability Index (PI).**AI Integration:**AI Tool: Google Sheets + NPV/IRR functions to evaluate capital investment alternatives. Monte Carlo simulation for capital budgeting risk analysis. Decision tree analysis for investment appraisal using scikit-learn..

Unit V: Financial Accounting and Analysis:

Content: Right Understanding, Right Feeling, Right Action: The value-ethics-skill triad; Professional Ethics: Codes of conduct for engineers (ASME, IEEE, IEI).Corruption, Dishonesty, and Short-term Thinking: Causes and value-based remedies; Vision for a Value-based Society; AI Ethics: Bias, fairness, transparency, and accountability.

AI Integration: Automated financial statement preparation using Python (pandas). AI Tool: ChatGPT to interpret company financial ratios. Project: ratio analysis of an Indian engineering company's annual report (BHEL, L&T, Infosys).

Text Books:

1. Mithani, D. M. (2025). Managerial Economics: Theory and Applications (10th ed.). Himalaya Publishing House.
2. Pandey, I. M. (2022). Financial Management (12th ed.). Vikas Publishing House.

Reference Books:

1. Aryasri, A. R. (2020). Managerial Economics and Financial Analysis (Revised ed.). McGraw-Hill Education.

2. Prasanna Chandra (2019). Financial Management: Theory and Practice (10th ed.). Tata McGraw-Hill.
3. Kumar, U. D. (2021). Business Analytics: The Science of Data-Driven Decision Making (2nd ed.). Wiley India.
4. Siddiqui, S. A. (2019). Engineering Economics and Financial Accounting (3rd ed.). S. Chand Publishing.

Online Learning Resources:

1. NPTEL (IIT): Managerial Economics – https://nptel.ac.in/courses/110105063
2. NPTEL (IIT): Financial Accounting for Engineers – https://nptel.ac.in/courses/110107114
3. Coursera: Economics for Non-Economists – https://www.coursera.org/learn/economics
4. Khan Academy: Microeconomics & Finance Basics
5. EdX- Financial Accounting (MIT OpenCourseWare)- https://www.edx.org/course/financial-accounting
6. SWAYAM Portal: Business Economics – https://swayam.gov.in

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26ES01201T	ES	Sustainability and Green Engineering (AI Integrated) (Common to all branches of Engineering)	2	0	0	0	2

Pre-Requisites: Basic knowledge of Chemistry, Environmental Science, Physics, and Mathematics, along with an understanding of sustainability concepts and engineering applications.

Course Objectives:

- To understand the concepts of sustainability, carbon cycle, and the environmental impact of engineering activities.
- To evaluate sustainable materials, their life cycle, and environmental performance.
- To apply energy calculations including embodied energy, operational energy, and life cycle energy.
- To assess green standards, energy codes, and performance rating systems (LEED, GRIHA).
- To integrate AI/ML tools for carbon footprint analysis, energy optimization, and sustainable design decision-making.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Analyze the principles of sustainability, Sustainable Development Goals (SDGs), carbon cycle, and greenhouse gas accounting to assess their interrelationships and environmental impacts. (L4)

CO2: Apply knowledge of sustainable materials to evaluate their environmental performance using life cycle assessment methods. (L3)

CO3: Analyze the embodied energy and operational energy of engineering systems using computational tools. (L4)

CO4: Evaluate green rating systems and assess compliance of design with energy codes using simulation and AI tools. (L4)

CO5: Design sustainable engineering solutions integrating circular economy principles and AI-driven optimization tools. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	1				3	2				2	1	1
CO2	2	2	2	2	2	2	2				2	2	1
CO3	2	2	2	2	3	2	1				2	2	2
CO4	2	2	3	2	3	3	2				3	1	1
CO5	2	3	3	3	3	3	2				3	2	2

Unit I: Fundamentals of Sustainability and Carbon Cycle:

Content: Definition of sustainability: three pillars (economic, social, environmental); Brundtland definition; Sustainable Development Goals (SDGs): SDGs 6, 7, 9, 11, 12, 13.Carbon cycle: biotic and abiotic components; Greenhouse gases: CO_2, CH_4, N_2O, HFCs; Carbon dioxide equivalent (CO_2e).Carbon footprint: definition, calculation methodology, carbon accounting; Climate change: global warming, sea level rise, extreme events.

AI Integration: AI for carbon footprint tracking: overview of ML-based life cycle assessment tools. Python-based carbon footprint calculator computing embodied carbon from material/process quantities. Free/Open-Source AI Tools: openLCA; Python (Pandas, Matplotlib) on Google Colab; Global Carbon Atlas.

Unit II: Sustainable Materials and Resource Selection:

Content: Sustainable materials and resource efficiency; Indoor air quality considerations; Recycled and alternative material sources; Durability and service life assessment.Life cycle of engineering materials; Multi-criteria decision analysis for material selection.

AI Integration: ML for sustainable material selection using multi-criteria decision analysis in Python. AI-based prediction of material performance using regression models (scikit-learn). Comparative LCA using openLCA. Free/Open-Source AI Tools: scikit-learn; openLCA; Python (NumPy, scikit-learn) on Google Colab.

Unit III: Embodied Energy and Operational Energy:

Content: Definition of embodied energy: extraction, processing, manufacturing, transport, construction, demolition; Operational energy: heating, cooling, lighting, equipment use. Life cycle energy: sum of embodied and operational energy; Energy concept: primary energy, renewable vs. non-renewable energy. Energy balance: reducing operational energy vs. increasing efficiency investment (embodied energy trade-off).

AI Integration: Python-based embodied energy calculator. AI for energy optimization: overview of EnergyPlus and OpenStudio simulations. ML for operational energy prediction (Linear Regression, Random Forest using scikit-learn). Free/Open-Source AI Tools: EnergyPlus; OpenStudio; Python (scikit-learn, NumPy) on Google Colab.

Unit IV: Green Rating Systems and Energy Codes:

Content: Energy Conservation and Sustainability Building Code (ECSBC-2023): scope, mandatory and prescriptive requirements; Green building rating systems: LEED, GRIHA, IGBC, BEE star rating.Passive design strategies: orientation, natural ventilation, daylighting; Net Zero Energy concepts: definition, pathways.

AI Integration: AI for automated LEED/GRIHA compliance checking. ML for performance prediction from design parameters. IoT sensors for real-time energy monitoring; AI-driven optimization case studies. Free/Open-Source AI Tools: EnergyPlus; OpenStudio; Python for data analysis on Google Colab.

Unit V: Circular Economy and Environmental Sustainability:

Content: Circular economy principles: reduce, reuse, recycle, recover, redesign; Waste hierarchy; Industrial waste utilization.Water recycling; Green procurement policies; Carbon sequestration potential of green infrastructure; Acid rain: causes, effects, prevention..

AI Integration: AI for waste prediction using ML models. Computer vision for automated waste sorting (overview). Python for waste audit data analysis. Sustainable infrastructure planning using AI and GIS. Free/Open-Source AI Tools: Python (scikit-learn, Pandas) on Google Colab; QGIS; openLCA; i-Tree.

Text Books:

1. Kibert, C. J. (2016). Sustainable Construction: Green Building Design & Delivery (4th ed.). Wiley.
2. Goodhew, S. (2016). Sustainable Construction Processes. Wiley-Blackwell.

Reference Books:

1. Langston, C. A., & Ding, G. K. C. (2011). Sustainable Practices in the Built Environment. Butterworth Heinemann.
2. ECSBC 2024 (Energy Conservation and Sustainability Building Code). Bureau of Energy Efficiency, Government of India.
3. Goswami, D., Kreith, F., & Kreider, J. (2000). Principles of Solar Engineering. Taylor & Francis.

Online Learning Resources:

1. NPTEL: Sustainable Development – https://nptel.ac.in/courses/105105157
2. NPTEL: Green Buildings – https://archive.nptel.ac.in/courses/105/102/105102195
3. Bureau of Energy Efficiency (BEE), India: https://beeindia.gov.in
4. IGBC (Indian Green Building Council): https://igbc.in
5. openLCA (free LCA software): https://www.openlca.org
6. EnergyPlus (free building energy simulation): https://energyplus.net
7. OpenStudio (free building energy modelling): https://openstudio.net
8. Global Carbon Project: https://www.globalcarbonproject.org

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC05205T	PC	**Intelligent Database Management Systems** (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML))	3	0	0	0	3

Pre-Requisites: Knowledge of computer fundamentals, programming concepts, data structures, and basic mathematical foundations including sets and relations.

Course Objectives:

- To understand fundamental concepts of database systems, data models, and DBMS architecture.
- To design relational databases using ER modelling and normalization techniques.
- To master SQL for data definition, manipulation, querying, and transaction control.
- To explore transaction management, concurrency control, and database recovery.
- To introduce modern database systems — NoSQL, vector databases — and AI-assisted data handling.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Explain DBMS architecture, data models, and ER modelling; design ER diagrams and map them to relational schemas. (L4)

CO2: Apply relational algebra and write SQL queries for data definition, manipulation, retrieval, and control. (L4)

CO3: Apply normalization techniques — 1NF through BCNF — to eliminate redundancy and design efficient relational schemas (L4)

CO4: Analyse transaction schedules for serializability; apply concurrency control protocols and recovery mechanisms. (L4)

CO5: Evaluate modern database systems including NoSQL and vector databases; apply AI-assisted querying and data governance. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	2	1		1				1	2	2
CO2	3	3	2	2	2		1				1	2	3
CO3	3	3	3	2	1		1				1	3	3
CO4	3	3	2	3	1		1				1	3	3
CO5	3	3	3	3	3		1				1	3	3

Unit I: Introduction to DBMS and Data Modelling:

Content: File system vs. DBMS: data redundancy, inconsistency, advantages of DBMS; 3-level DBMS architecture (internal, conceptual, external); logical and physical data independence; data models — hierarchical, network, relational, object-oriented.ER Model: entities (strong and weak), attributes (simple, composite, multivalued, derived), relationships; keys (primary, candidate, composite); cardinality ratios (1:1, 1:N, M:N) and participation constraints (total, partial); ER diagrams.ER-to-relational mapping: rules for entities, weak entities, binary relationships, multivalued attributes; introduction to centralized vs. distributed and cloud database systems; overview of NoSQL categories.

AI Integration: Use draw.io and MySQL Workbench to design ER diagrams; use ChatGPT to convert natural language problem statements into ER models and validate the output manually.

Unit II: Relational Model and SQL:

Content: Relational model: relations, tuples, attributes, domains, schemas; relational algebra — selection (σ), projection (π), union, set difference, Cartesian product, natural join, outer joins, intersection.SQL: DDL (CREATE, ALTER, DROP); DML (INSERT, UPDATE, DELETE); DQL (SELECT with WHERE, ORDER BY, GROUP BY, HAVING, aggregate functions, joins, subqueries, set operations); DCL (GRANT, REVOKE); TCL (COMMIT, ROLLBACK, SAVEPOINT).Views, indexes, and query optimization: simple and complex views; B-Tree and B+ Tree indexing — dense vs. sparse; primary and secondary indexes; query optimization basics — cost-based optimizer, predicate pushdown.

AI Integration: Use MySQL and SQLite for SQL query implementation; use ChatGPT to generate SQL from natural language and verify manually; use GitHub Copilot to scaffold complex queries; use Google Colab for SQL exercises.

Unit III: Database Design and Normalization:

Content: Functional dependencies (FDs): trivial and non-trivial FDs; Armstrong's axioms; closure of FD sets; canonical (minimal) cover; types of dependencies — partial, transitive, multivalued.Normalization: 1NF (atomic values, no repeating groups), 2NF (no partial dependency), 3NF (no transitive dependency), BCNF (every determinant is a superkey); introduction to 4NF and 5NF.Decomposition: lossless join decomposition; dependency preservation; BCNF decomposition algorithm; denormalization trade-offs for OLAP workloads.

AI Integration: Use ChatGPT to input a schema and get normalized tables; validate normalization steps manually; use Jupyter Notebook for FD closure calculations; practice AI-assisted schema design and verify with Armstrong's axioms.

Unit IV: Transaction Management and Concurrency Control:

Content: Transaction concepts: ACID properties (Atomicity, Consistency, Isolation, Durability); transaction states; COMMIT, ROLLBACK, SAVEPOINT; isolation levels — READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE.Concurrency control: lock-based protocols — shared and exclusive locks, two-phase locking (2PL), strict 2PL; timestamp-based protocols; deadlock detection (waits-for graph), prevention (wound-wait, wait-die), avoidance.Serializability: conflict serializability; precedence graph construction; view serializability. Recovery: log-based recovery — undo and redo; Write-Ahead Log (WAL); checkpointing; ARIES algorithm (Analysis, Redo, Undo).

AI Integration: Use Python simulations to create concurrent transaction schedules; use ChatGPT to analyse serializability and trace precedence graph construction; simulate lost update and dirty read scenarios and resolve.

Unit V: Intelligent Data Management for AI Systems:

Content: NoSQL systems: motivation (3Vs, schema flexibility, horizontal scale); CAP theorem; document stores (MongoDB — aggregation pipeline), key-value (Redis — TTL, sorted sets), column-family (Cassandra — partition key), graph (Neo4j — Cypher).Text-to-SQL and LLM-over-database: natural language to SQL translation; schema-linking; accuracy challenges; RAG-over-SQL (retrieve rows → LLM context → answer); LLM as data analyst; verification requirements before production use. Vector data management: vector databases (FAISS, Pinecone, Weaviate, Chroma); ANN search (HNSW, IVF); similarity metrics (cosine, Euclidean, dot product); hybrid search (vector + scalar filter); re-ranking. Data governance: data catalog (Apache Atlas), lineage tracking; GDPR Article 5; India DPDP Act 2023.

AI Integration: Use ChatGPT to generate SQL from natural language and verify each query; use Claude to compare HNSW vs. IVF for a vector search workload; use MongoDB Atlas for document store experiments; use an AI assistant to generate a data governance checklist.

Text Books:

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Database System Concepts, 7th Edition, McGraw-Hill, 2019.
2. Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson Education, 2016.

Reference Books:

1. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill.
2. Martin Kleppmann, Designing Data-Intensive Applications, O'Reilly Media, 2017.

Online Learning Resources:

1. NPTEL – Database Management Systems, IIT Madras – nptel.ac.in
2. MySQL Documentation – dev.mysql.com/doc | MongoDB – docs.mongodb.com
3. Stanford – Introduction to Databases – online.stanford.edu
4. India DPDP Act 2023 – meity.gov.in | FAISS – github.com/facebookresearch/faiss

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC31202T	PC	Intelligent Data Processing and System Integration	3	0	0	0	3

Pre-Requisites: Python Programming, Data Structures, Databases (DBMS)and Operating Systems **.**

Course Objectives:

- To introduce data ingestion patterns — batch, streaming, ETL/ELT — and system integration techniques using REST APIs, Kafka, and integration design patterns.
- To develop competency in intelligent data processing using Pandas, NumPy, and Apache Spark for scalable, production-grade AI data pipelines.
- To build knowledge of database integration for AI systems — SQL/NoSQL from Python, ORM, vector databases, and data access patterns for AI APIs.
- To introduce REST API design and microservice integration using FastAPI and gRPC for building AI-powered services.
- To develop understanding of workflow orchestration with Apache Airflow, pipeline monitoring, error handling, and responsible data integration under GDPR and India DPDP Act 2023.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Explain batch and streaming data ingestion patterns; design ETL/ELT pipelines using connectors, APIs, and message queues. (L4)

CO2: Apply Pandas and NumPy for intelligent data processing; implement Spark DataFrames for distributed transformation. (L3)

CO3: Integrate SQL, NoSQL, and vector databases with Python applications using SQLAlchemy, pymongo, and FAISS. (L3)

CO4: Design and implement REST APIs and gRPC services using FastAPI for AI system integration. (L4)

CO5: Orchestrate multi-step data workflows using Apache Airflow; apply logging, monitoring, and GDPR/DPDP compliance to integrated AI systems. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	2	2	1	1				1	3	3
CO2	3	3	2	2	3		1				1	3	3
CO3	3	3	2	2	3		1				1	3	3
CO4	3	3	3	2	3	1	1				1	3	3
CO5	3	3	3	3	3	2	1				1	3	3

Unit I: Data Ingestion and System Integration Patterns:

Content: Data ingestion patterns: batch ingestion (scheduled, full-load vs. incremental, change data capture — CDC using Debezium); streaming ingestion (event-driven, real-time); micro-batch (Spark Structured Streaming); comparing latency, throughput, and complexity of each pattern.ETL vs. ELT: Extract-Transform-Load (transform before loading — traditional data warehouse); Extract-Load-Transform (load raw, transform in-place — modern cloud data stack using dbt); when to use each; EL+T hybrid pattern for AI feature pipelines.System integration techniques: REST API integration (requests library — GET, POST, PUT, DELETE, authentication, pagination, retry with backoff); webhook-based integration; file-based integration (SFTP, S3-compatible object storage); message queue integration (Apache Kafka — producers and consumers from Python using kafka-python).Integration patterns: adapter pattern (wrap external API in a standard internal interface); façade pattern (simplify a complex subsystem); event-driven integration (publish events on state changes, subscribe to external events); strangler fig pattern (incremental replacement of legacy systems with AI-powered components).

AI Integration: Use ChatGPT to explain CDC vs. full-load batch ingestion with a concrete e-commerce database example; use Claude to design an integration pattern for connecting a legacy ERP system to an AI prediction service; use an AI assistant to generate retry-with-backoff code for a REST API integration.

Unit II: Intelligent Data Processing:

Content: NumPy for intelligent processing: vectorised operations vs. loops (performance comparison); broadcasting for batch computations; linear algebra (dot, matmul, linalg.solve) for AI feature computation; array manipulation (reshape, stack, split, where, argsort); memory-efficient dtypes (float32 vs. float64 for AI workloads).Pandas for intelligent data processing: method chaining with pipe(); assign() for derived columns; query() for readable filtering; apply() with custom functions; groupby with transform() for group-level enrichment without reducing rows; merge_asof() for time-series joins.Data quality in processing pipelines: inline validation using assert statements and custom functions; detecting and handling schema drift; data freshness checks; building a processing audit log (input row count, output row count, null rate before/after, processing time).Apache Spark for distributed processing: SparkSession, DataFrames, lazy evaluation DAG; transformations (select, filter, withColumn, groupBy, join, window functions) vs. actions (show, count, collect, write); Spark SQL; UDFs for custom AI logic; reading from and writing to Parquet, Delta Lake, and JDBC.

AI Integration: Use ChatGPT to explain Spark lazy evaluation with a DAG example; use Claude to identify performance bottlenecks in a given Pandas processing pipeline and suggest vectorised alternatives; use GitHub Copilot to implement a Spark UDF for a custom text processing step.

Unit III: Database Integration For Ai Systems: Content: SQL database integration with Python: SQLAlchemy Core (connection, execute, text()) and ORM (declarative base, session, model classes, relationships); connection pooling — QueuePool, NullPool; bulk insert (bulk_insert_mappings), upsert (on_conflict_do_update for PostgreSQL); Alembic for database migrations.NoSQL integration: pymongo — connecting to MongoDB, CRUD operations, aggregation pipeline from Python, indexing; Redis integration with redis-py — key-value storage, TTL, sorted sets; Cassandra with cassandra-driver — session, prepared statements, batch operations; choosing NoSQL for AI: document store for ML experiment metadata, key-value for feature serving, column-family for time-series sensor data.Vector database integration: FAISS from Python — IndexFlatL2, IndexIVFFlat, index serialisation; Chroma — PersistentClient, collection.add(), collection.query(); Pinecone client — upsert, query with metadata filter; pattern: embed text/image → store vector+metadata → query by similarity; hybrid search (vector + scalar filter).Data access patterns for AI systems: repository pattern (abstract data access behind an interface); CQRS (Command Query

Responsibility Segregation — separate read and write models for high-throughput AI APIs); read-through and write-through caching with Redis; connection pool tuning for concurrent AI inference requests.
AI Integration: Use ChatGPT to explain the SQLAlchemy ORM session lifecycle; use Claude to design a repository pattern for an AI application that needs to switch between SQLite (development) and PostgreSQL (production); use an AI assistant to explain FAISS index types and when to use each..

Unit IV: API Design and Microservice Integration:
Content: REST API design principles: resource naming conventions, HTTP methods (GET, POST, PUT, PATCH, DELETE) and their semantics, status codes (2xx, 4xx, 5xx), idempotency, statelessness; versioning strategies (URL versioning /v1/, header versioning); pagination (cursor-based vs. offset-based); OpenAPI specification (Swagger) for API documentation.FastAPI for AI system integration: path operations, request/response models with Pydantic, path and query parameters, request body validation; dependency injection for database sessions and model loading; background tasks for async processing; middleware for logging, CORS, and rate limiting; lifespan events for model warm-up on startup.Asynchronous processing and inter-service communication: async/await in Python for non-blocking I/O; asyncio event loop; aiohttp for async HTTP clients; gRPC from Python — protobuf schema definition, code generation, unary and server-streaming RPCs; gRPC vs. REST for internal AI microservice communication.AI API integration patterns: model-as-a-service (expose ML model via REST API); circuit breaker pattern (fail fast when downstream service is unavailable using tenacity or circuitbreaker library); API gateway pattern (single entry point for multiple AI services — authentication, rate limiting, routing); webhook callbacks for async AI job completion.
AI Integration: Use ChatGPT to design a FastAPI endpoint for a text classification model with input validation and background logging; use Claude to explain the circuit breaker pattern with a concrete AI microservice failure scenario; use GitHub Copilot to generate a gRPCprotobuf schema for a model prediction service

Unit V: Workflow Orchestration and Responsible Integration:
Content: Apache Airflow for workflow orchestration: DAG (Directed Acyclic Graph) structure — tasks, operators, dependencies; core operators — PythonOperator, BashOperator, BranchPythonOperator, ShortCircuitOperator; scheduling (cron expressions, timedelta); XCom for inter-task communication; task retries, retry_delay, timeout; Airflow connections and variables for credential management.Pipeline monitoring and observability: structured logging (Python logging module with JSON formatter); metrics collection (Prometheus client library — counters, gauges, histograms); distributed tracing (OpenTelemetry — trace IDs across service calls); alerting on pipeline failures (Airflow email/Slack callbacks, Prometheus Alertmanager); data quality monitoring (Great Expectations checkpoints in Airflow DAGs).Error handling and resilience in integrated systems: exception hierarchy design for pipeline errors (DataIngestionError, TransformationError, IntegrationError); dead-letter queues (DLQ) for unprocessable messages; idempotent processing (safe to re-run without side effects); saga pattern for distributed transactions across multiple databases/services; graceful degradation (return cached result when live inference fails).Responsible system integration: PII detection in data pipelines (using spaCy NER or regex to flag names, emails, phone numbers, Aadhaar, PAN before persistence); data minimisation in integration design; consent-aware data routing (route data tagged with consent flags to appropriate storage); GDPR Article 5 and India DPDP Act 2023 applied to integrated AI systems — data subject rights (access, erasure) in a multi-database architecture; audit logging for compliance.

AI Integration: Use ChatGPT to explain Airflow XCom with a concrete multi-task DAG example; use Claude to design a responsible data routing pipeline that respects consent flags under India DPDP Act 2023; use an AI assistant to generate a structured logging configuration with JSON formatter for a FastAPI application.

Text Books:

1. Martin Kleppmann, Designing Data-Intensive Applications, O'Reilly Media, 2017. (Primary — Units I, II, V)
2. Wes McKinney, Python for Data Analysis, 3rd Edition, O'Reilly Media, 2022. (Unit II — NumPy, Pandas)
3. Chip Huyen, Designing Machine Learning Systems, O'Reilly Media, 2022. (Units I, III, IV, V)
4. Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson. (Unit III)

Reference Books:

1. Sam Newman, Building Microservices, 2nd Edition, O'Reilly Media, 2021.
2. Bas Geerdink, A Practical Guide to Building Ethical AI Systems, Manning, 2023.

Online Learning Resources:

1. Apache Airflow – airflow.apache.org |FastAPI – fastapi.tiangolo.com
2. SQLAlchemy – sqlalchemy.org | Apache Spark – spark.apache.org
3. FAISS – github.com/facebookresearch/faiss | Chroma – trychroma.com
4. Great Expectations – greatexpectations.io |OpenTelemetry Python – opentelemetry.io
5. NPTEL – Data Science for Engineers, IIT Madras – nptel.ac.in
6. India DPDP Act 2023 – meity.gov.in | GDPR – gdpr.eu

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC31203T	PC	Computer Architecture for Intelligent Systems	2	0	2	0	3

Pre-Requisites: Digital Logic Design, Python Programming, Data Structures and Operating Systems (basic).

Course Objectives:

- To introduce computer organization fundamentals — ISA, RISC-V, data representation — and explain how they shape AI software performance.
- To develop understanding of CPU microarchitecture — pipelining, hazards, out-of-order execution, and SIMD — as the foundation for AI workload throughput.
- To build knowledge of memory systems — cache hierarchy, virtual memory, NUMA — and explain the memory wall's impact on AI workloads.
- To introduce I/O subsystems, storage hierarchy, and PCIe interconnects as the infrastructure connecting CPUs to AI accelerators.
- To develop understanding of OS-level mechanisms — scheduling, NUMA-aware allocation, containers, power management — for AI workload performance optimisation.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Apply ISA design principles, RISC and CISC trade-offs, RISC-V instruction encoding, and number representation techniques to solve problems involving AI data types. (L3)

CO2: Describe CPU pipelining, hazards, out-of-order execution, and SIMD/AVX; analyse how these mechanisms affect AI workload throughput. (L4)

CO3: Explain cache hierarchy, locality principles, virtual memory, and NUMA; identify memory bottlenecks in AI algorithms using the roofline model. (L4)

CO4: Apply concepts of DMA, PCIe, NVMe storage, hardware accelerator attachment, and storage hierarchy to select suitable storage solutions for ML datasets. (L3)

CO5: Explain OS scheduling, NUMA-aware memory allocation, containerization, and power management for AI workload performance optimization. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	1	2	2		1				1	3	3
CO2	3	3	2	3	2		1				1	3	3
CO3	3	3	2	3	2		1				1	3	3
CO4	3	2	1	2	2		1				1	3	3
CO5	3	3	2	2	2		1				1	3	3

Unit I: Computer Organization And ISA:

Content: Computer organization: Von Neumann architecture — fetch-decode-execute cycle, registers, ALU, control unit, memory bus; Harvard architecture — separate instruction and data memories (used in microcontrollers and edge AI chips); stored-program concept; performance equation: CPU time = instruction count × CPI × clock cycle time.

Number representation for AI: signed integers (two's complement); IEEE 754 floating-point — float32 (23-bit mantissa), float16 (10-bit mantissa, 5-bit exponent), bfloat16 (8-bit mantissa, 8-bit exponent — used in TPUs and modern GPUs for AI training); precision trade-offs — float32 for training, float16/bfloat16 for inference; overflow, underflow, NaN, infinity handling.Instruction Set Architecture (ISA): ISA as the hardware-software interface; CISC (x86 — complex variable-length instructions, memory-to-memory operations) vs. RISC (ARM, RISC-V — fixed-length, load-store, simple instructions); RISC-V ISA — base integer ISA (RV32I, RV64I), extensions (M: multiply/divide, F: single-precision float, D: double-precision, V: vector for AI workloads); RISC-V instruction formats (R, I, S, B, U, J).Amdahl's Law and performance analysis: Amdahl's Law — speedup limited by the non-parallelizable fraction; application to AI: if 90% of a training loop is matrix multiply and we accelerate it 100×, total speedup ≤ 10×; strong scaling vs. weak scaling; FLOPS as an AI performance metric; TFLOPS of modern AI hardware.

AI Integration: Use ChatGPT to derive Amdahl's Law speedup for a given AI training loop breakdown; use Claude to explain bfloat16 vs. float16 precision trade-offs for a transformer training scenario; use an AI assistant to decode a RISC-V instruction encoding in binary

Unit II: CPU Microarchitecture for AI Workloads:

Content: Pipelining: five-stage RISC-V pipeline (IF, ID, EX, MEM, WB); pipeline CPI = 1 ideally; structural hazards (resource conflicts — solution: stall or duplicate hardware); data hazards — RAW (Read After Write) — forwarding/bypassing paths; WAW and WAR hazards; control hazards — branch prediction (static: predict not-taken; dynamic: 2-bit saturating counter, branch history table, branch target buffer).Out-of-order execution and superscalar: in-order vs. out-of-order execution; Tomasulo's algorithm — reservation stations, common data bus (CDB), register renaming to eliminate WAW/WAR hazards; reorder buffer (ROB) for precise exceptions; superscalar execution — issuing multiple instructions per cycle; issue width and IPC; modern x86 and ARM cores (4–8 wide issue).SIMD and vector processing for AI: SIMD (Single Instruction Multiple Data) — same operation on multiple data elements simultaneously; x86 SIMD extensions — MMX (64-bit), SSE/SSE2 (128-bit), AVX/AVX2 (256-bit), AVX-512 (512-bit); Python NumPy and PyTorch automatically use AVX-512 via optimized BLAS libraries; SIMD for matrix multiply: vectorised dot product of 16 float32 values in one AVX-512 instruction; RISC-V V extension.CPU performance analysis for AI: CPI breakdown — compute-bound vs. memory-bound operations; roofline model — arithmetic intensity (FLOP/byte), compute ceiling (peak FLOPS), bandwidth ceiling (peak GB/s × arithmetic intensity); identifying whether a given AI layer (attention, convolution, fully-connected) is compute-bound or memory-bound; Intel VTune and perf tool concepts for CPU profiling.

AI Integration: Use ChatGPT to trace Tomasulo's algorithm for a 5-instruction sequence with RAW hazards; use Claude to apply the roofline model to a transformer self-attention layer — determine if it is compute-bound or memory-bound on a given CPU; use an AI assistant to explain why NumPy matrix multiply is faster than a Python loop (BLAS + SIMD)..

Unit III: Memory Systems and the Memory Wall:

Content: Cache hierarchy: SRAM cache vs. DRAM main memory — speed, cost, capacity trade-offs; direct-mapped, set-associative (2-way, 4-way, 8-way), and fully-associative caches; cache parameters — cache size, block size, associativity; cache miss types — compulsory (cold), capacity, conflict; replacement policies — LRU, pseudo-LRU, random; write policies — write-through vs. write-back with write-allocate.Cache performance and locality: average memory access time (AMAT = hit time + miss rate × miss penalty); multilevel cache (L1, L2, L3) AMAT formula; spatial locality (accessing nearby memory) and temporal locality (reusing same data); cache-oblivious algorithms; how cache blocking (loop tiling) improves AI training performance.

Virtual memory and memory management: virtual address space; page table; TLB (Translation Lookaside Buffer) — TLB hit vs. miss; page fault; page replacement policies (LRU, clock algorithm); huge pages (2 MB vs. 4 KB) — reducing TLB pressure for large ML model weight tensors; memory-mapped files (mmap) for large dataset loading used by PyTorchDataLoader.

DRAM architecture and the memory wall: DRAM cell, row/column addressing, RAS/CAS timing; DRAM bandwidth (GB/s) vs. DRAM latency (ns); DDR4/DDR5 specifications; NUMA (Non-Uniform Memory Access) — multi-socket systems, local vs. remote DRAM access time; memory bandwidth as the bottleneck for LLM inference (KV-cache is memory-bound); roofline analysis: transformer inference is memory-bound, training is compute-bound.

AI Integration: Use ChatGPT to compute AMAT for a given two-level cache hierarchy with miss rates; use Claude to explain how loop tiling improves matrix multiply cache performance with a concrete 4×4 tile example; use an AI assistant to explain why LLM inference is memory-bound using the roofline model for a given model size and hardware bandwidth.

Unit IV: I/O, Storage, and Interconnects for AI Systems:

Content: I/O subsystem: polling vs. interrupt-driven I/O vs. DMA (Direct Memory Access) — DMA controller transfers data between I/O device and DRAM without CPU involvement; IOMMU (I/O Memory Management Unit) for secure DMA from accelerators; PCIe generations (PCIe 3.0: 8 GB/s, PCIe 4.0: 16 GB/s, PCIe 5 .0: 32 GB/s per ×16 slot) and their impact on GPU-to-CPU data transfer bandwidth.Storage hierarchy for AI systems: register → L1/L2/L3 cache → DRAM → NVMe SSD → SATA SSD → HDD → network storage (S3, NFS); latency and bandwidth at each level; NVMe SSD (PCIe 4.0 ×4 — up to 7 GB/s sequential read, ~100 μs latency) vs. SATA SSD (600 MB/s) vs. HDD (150 MB/s, 10 ms seek); GPUDirect Storage — data path from NVMe directly to GPU VRAM bypassing CPU DRAM.Hardware accelerator attachment model: PCIe device enumeration and BAR (Base Address Register) mapping; CPU-GPU memory model — host (DRAM) vs. device (VRAM) memory; cudaMalloc, cudaMemcpy (H2D, D2H, D2D); unified memory (CUDA UM); pinned (page-locked) host memory — faster H2D transfer; PCIe bandwidth as the bottleneck for small batch inference.Profiling and performance counters: hardware performance counters — CPU cycle count, cache miss count, branch misprediction count, instructions retired; Intel PMU; perf stat for Linux profiling; top-down microarchitecture analysis — frontend bound, backend bound, bad speculation, retiring; energy efficiency metrics — GFLOPS/W, performance per watt.

AI Integration: Use ChatGPT to explain DMA data transfer for a GPU training step — trace bytes from NVMe SSD through DRAM to GPU VRAM; use Claude to explain PCIe bandwidth bottleneck for small batch GPU inference; use an AI assistant to interpret a perf stat output for a NumPy matrix multiply — identify the dominant performance bottleneck.

Unit V: Operating Systems and AI Workload Management:

Content: Process and thread model for AI: process (independent address space, OS-scheduled), thread (shared address space, lighter weight), coroutine/async (cooperative, no preemption — used by Python asyncio); Python Global Interpreter Lock (GIL) — prevents true CPU parallelism in CPython for pure Python code; GIL release for C-extension calls (NumPy, PyTorch); multiprocessing vs. multithreading for data loading in PyTorch (num_workers).CPU scheduling for AI workloads: preemptive scheduling (FCFS, SJF, Round Robin, priority scheduling); real-time scheduling (EDF — Earliest Deadline First) for latency-sensitive inference; CPU affinity — pinning a process to specific cores to reduce cache invalidation (taskset); NUMA-aware scheduling — schedule threads on cores local to the NUMA node holding their data; cgroups for resource isolation of AI containers.

Memory management for AI: virtual memory recap; memory allocators — glibc malloc, jemalloc, tcmalloc (used by TensorFlow); memory fragmentation and its impact on long-running AI training jobs; huge pages (transparent huge pages — THP) for large tensor allocations; NUMA-aware allocation (numactl --membind, --cpunodebind); copy-on-write (COW) for forked DataLoader worker processes in PyTorch.

Containers and virtualization for AI: OS virtualization — hypervisors (Type 1: VMware ESXi, KVM; Type 2: VirtualBox); containerization — cgroups + namespaces; Docker for AI model packaging; Kubernetes for AI service orchestration — Pod, Deployment, Service, resource limits (requests/limits for CPU, memory, nvidia.com/gpu); power management — CPU frequency scaling (intel_pstate governor), thermal throttling impact on sustained AI workloads.

AI Integration: Use ChatGPT to explain the Python GIL and why PyTorchDataLoader workers use multiprocessing instead of multithreading; use Claude to explain NUMA-aware memory allocation for a 2-socket server running a large language model inference service; use an AI assistant to write a Dockerfile for a FastAPI-based AI inference service.

Text Books:

1. David A. Patterson and John L. Hennessy, Computer Organization and Design (RISC-V Edition), Morgan Kaufmann.
2. John L. Hennessy and David A. Patterson, Computer Architecture: A Quantitative Approach, 6th Edition, Morgan Kaufmann.
3. Vijay Janapa Reddi et al., ML Systems: Principles and Practices (free at mlsysbook.ai).
4. Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Operating System Concepts, 10th Edition, Wiley.

Reference Books:

1. William Stallings, Computer Organization and Architecture, 11th Edition, Pearson Education.
2. Onur Mutlu, Computer Architecture Lectures, Carnegie Mellon University (free on YouTube).

Online Learning Resources:

1. NPTEL – Computer Architecture, IIT Madras – nptel.ac.in
2. MIT 6.004 Computation Structures – ocw.mit.edu
3. RISC-V Specifications – riscv.org/technical/specifications
4. Intel x86 Intrinsics Guide (SIMD/AVX) – software.intel.com/sites/landingpage/IntrinsicsGuide
5. MLSys Book – mlsysbook.ai | Perf Wiki – perf.wiki.kernel.org

B. Tech. – II Year I Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC31202P	PC	Intelligent Data Processing and System Integration Lab	0	0	0	3	1.5

Pre-Requisites: Knowledge of Python programming, databases, data structures, computer networks, and basic data analytics concepts.

Course Objectives:

- To understand and apply data integration, intelligent data processing, database connectivity, API development, and distributed computing concepts for building scalable AI-enabled applications.
- To design and develop integrated data pipelines, microservices, vector databases, APIs, and workflow orchestration systems using modern software tools and frameworks.
- To apply AI-assisted development, performance optimization, observability, security, privacy, and responsible data management practices in real-world application development.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Implement REST API integration with retry-backoff logic and Kafka-based streaming ingestion; build an ETL and ELT pipeline using Python. (L3)

CO2: Build intelligent Pandas processing pipelines with method chaining and validation; implement Spark DataFrame transformations on a large dataset. (L4)

CO3: Integrate Postgre SQL using SQL Alchemy ORM, MongoDB using pymongo, and FAISS vector index from Python; implement a hybrid search pipeline. (L3)

CO4: Design and implement Fast API REST endpoints and agRPC service ; apply circuit breaker and API gateway patterns for AI microservice integration. (L4)

CO5: Build and schedule Apache Airflow DAGs with branching and XCom ; instrument a pipeline with structured logging, Prometheus metrics, and GDPR/DPDP compliance. (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	2	2	3		3	3	3		1	2	3
CO2	3	3	2	2	3		3	3	3		1	2	3
CO3	3	3	2	2	3		3	3	3		1	1	3
CO4	3	3	3	2	3		3	3	3		1	3	3
CO5	3	3	3	3	3		3	3	3		1	3	3

List of Experiments:

Unit I: Data Ingestion and System Integration Patterns:

1. REST API Integration with Retry-Backoff and Webhook Receiver

Implement a REST API client using the requests library to consume a public API (e.g., OpenWeatherMap or JSONPlaceholder). Requirements: (a) GET with query parameters and response pagination — fetch all pages and aggregate results; (b) POST with JSON body and Bearer token authentication; (c) exponential retry-with-backoff using the tenacity library — configure max_attempts=5, wait_exponential(min=1, max=30), retry on requests.exceptions.ConnectionError and HTTP 429/503; (d) implement a simple webhook receiver using Flask — expose a POST /webhook endpoint, log incoming payloads to a JSON

file, and return HTTP 200. Test by sending 10 simulated webhook calls using requests.post(). Use ChatGPT to generate the retry decorator configuration — verify that it correctly retries on status 429 and document the AI output vs. your implementation.

2. Kafka Producer-Consumer Pipeline and ETL vs. ELT Comparison

Set up Apache Kafka locally (using Docker: docker-compose with Zookeeper + Kafka). Part A — Kafka Producer-Consumer: (a) write a Python producer (kafka-python) that publishes 100 JSON messages to a topic orders (each: {order_id, product, amount, timestamp}); (b) write a consumer that reads from orders, deserializes each message, and appends to a Parquet file using PyArrow; (c) demonstrate consumer group behaviour by running 2 consumers in the same group — show message partitioning. Part B — ETL vs. ELT: (a) implement ETL: extract from a CSV, transform (clean nulls, normalise strings, derive a new column), load into SQLite; (b) implement ELT: load the raw CSV into DuckDB as-is, then apply a dbt-style SQL transformation inside DuckDB; (c) measure and compare total time for both approaches on a 100K-row dataset. Use Claude to compare the two pipelines — record the response, evaluate it, and document one correction if needed.

3. File-Based Integration and S3-Compatible Object Storage

Part A — SFTP simulation: use paramiko to connect to a local SFTP server (sftpserver Docker image); upload 5 CSV files, list the remote directory, download a specific file, and handle FileNotFoundError gracefully. Part B — S3-compatible object storage using MinIO (Docker): (a) set up MinIO locally; (b) use boto3 to: create a bucket, upload a 10 MB Parquet file, list bucket contents with metadata, download the file, and delete it; (c) implement multipart upload for a 100 MB file using boto3 multipart API; (d) enable server-side encryption (SSE-S3) for the upload. Part C — Integration pattern: implement the adapter pattern — write a StorageAdapter abstract class with upload(), download(), list() methods; implement SFTPAdapter and S3Adapter that satisfy the same interface; demonstrate that switching from SFTP to S3 requires only changing the adapter without touching the calling code. Use an AI assistant to generate the abstract adapter class — review and implement it.

Unit II: Intelligent Data Processing:

4. NumPy Vectorised Processing and Performance Benchmarking

Benchmark vectorised vs. loop implementations across 5 operations on arrays of sizes [10K, 100K, 1M, 10M]: (a) element-wise square root; (b) computing row-wise dot product between two 2D arrays; (c) batch normalisation: (X − mean) / std using broadcasting; (d) computing a cosine similarity matrix between 1000 vectors of dimension 128 using np.dot and manual loop — compare times; (e) solving a linear system Ax = b for 100 different right-hand sides using np.linalg.solve vs. loop. Plot time vs. array size for all 5 on a single Matplotlib figure with log scale. Demonstrate memory-efficient dtype selection: compute mean absolute error between float32 and float64 results for the batch normalisation; report memory usage with arr.nbytes. Use ChatGPT to explain why NumPy broadcasting avoids the loop — verify with 1 counter-example where broadcasting raises an error, and document.

5. Pandas Intelligent Pipeline with Method Chaining, Validation, and Audit Log

Load the NYC Taxi Trip dataset (or equivalent with 500K+ rows). Build a fully method-chained Pandas pipeline using .pipe(): Stage 1 — clean: drop exact duplicates, clip trip_distance to [0.1, 100], fill null passenger_count with median; Stage 2 — validate: assert no nulls in required columns, assert all fare_amount> 0, raise DataQualityError if >5% rows fail; Stage 3 — enrich:

add hour_of_day, day_of_week, is_weekend, log1p_fare_amount, haversine_distance (from pickup/dropofflat-lon); Stage 4 — aggregate: groupbypickup_hour and passenger_count with multiple aggregations (mean, median, count, std) using pd.NamedAgg. Build a processing audit log: record input row count, output row count, null rate per column before/after, processing time per stage, and write to JSON. Apply merge_asof() for a time-series join: merge the trip dataset with a weather_dataDataFrame (temperature per 30-minute interval) on timestamp. Use Claude to identify 2 performance bottlenecks in the pipeline — implement the suggested optimisations, measure improvement, and document.

6. Apache Spark — Distributed DataFrame Processing and UDFs

Initialise a local SparkSession with 4 worker threads. Load a 1M-row CSV (generated with Faker or downloaded) into a Spark DataFrame. Demonstrate lazy evaluation: build a transformation chain of 5 operations (filter, withColumn, groupBy, join, sort) without calling an action; use df.explain() to print the physical plan; observe the DAG in the Spark UI. Transformations: (a) filter trips where distance > 2; (b) withColumn to add fare_per_km = fare / distance; (c) groupBy hour and compute mean fare, count, and percentile_approx(fare, 0.95) using Window; (d) join with a zones lookup DataFrame on location_id; (e) write output to Parquet with partitionBy('hour'). Implement a Spark UDF for custom text processing: clean address strings (lowercase, remove punctuation, strip) — compare UDF performance with a built-in Spark SQL function equivalent. Run the same aggregation using Spark SQL syntax (spark.sql(...)). Use ChatGPT to explain the difference between a narrow transformation and a wide transformation — classify each of your 5 transformations and verify against the explain() output.

Unit III: Database Integration for AI Systems:

7. SQLAlchemy ORM — Full CRUD, Connection Pooling, and Migrations

Design a data model for an ML experiment tracker: Experiment (id, name, created_at), Run (id, experiment_id FK, status, started_at, ended_at), Metric (id, run_id FK, key, value, step), Artifact (id, run_id FK, path, size_bytes). Implement using SQLAlchemy ORM (declarative base, relationship, backref) targeting PostgreSQL (use Docker for a local instance). Operations: (a) insert 5 experiments each with 3 runs and 10 metrics per run using bulk_insert_mappings for performance; (b) query: find the top-3 runs by max accuracy metric using joined query with subquery; (c) upsert: use on_conflict_do_update (PostgreSQL INSERT ... ON CONFLICT) to update an existing run's status; (d) configure QueuePool (pool_size=5, max_overflow=10, pool_timeout=30) and demonstrate concurrent inserts from 5 threads; (e) implement Alembic migrations: add a tags column to Experiment, generate a migration script with alembic revision --autogenerate, apply with alembic upgrade head, verify. Use Claude to design the repository pattern for this schema — implement a RunRepository class with 4 methods.

8. MongoDB, Redis, and NoSQL Integration for AI Data Patterns

MongoDB (pymongo): (a) design a document schema for AI model metadata: {model_id, name, framework, hyperparameters: {}, metrics: {}, tags: [], created_at}; insert 30 documents; (b) aggregation pipeline: group by framework, compute avg accuracy, max f1; (c) text index on name + tags, demonstrate $text search; (d) upsert 10 documents using update_one with upsert=True; (e) demonstrate change streams: open a change stream watch cursor, insert 3 documents in a separate thread, print the change events. Redis (redis-py): (a) key-value: cache ML model metadata with TTL=3600; implement cache-aside pattern — check cache, miss → query MongoDB, set cache; (b) sorted set: implement a leaderboard of top models by F1 score using ZADD/ZREVRANGE; (c) Pub/Sub: publish 10 training progress events on channel

ml:progress, consume in a subscriber thread and print. Use ChatGPT to explain the MongoDB aggregation pipeline stages $group, $project, $match — verify all 3 claims against your experimental output and document discrepancies.

9. Vector Database Integration — FAISS, Chroma, and Hybrid Search

Generate 1000 product descriptions; encode using sentence-transformers (all-MiniLM-L6-v2) to get 384-dim embeddings. FAISS: (a) build IndexFlatL2 and IndexIVFFlat (nlist=20); (b) for 20 queries measure Recall@5 and latency for both; (c) serialise the IVF index to disk (faiss.write_index) and reload (faiss.read_index) — verify results are identical; (d) implement a re-ranking step: FAISS retrieves top-20, then re-rank by BM25 score on product title. Chroma: (a) create a persistent collection, add 1000 embeddings with metadata (category, price, in_stock); (b) vector query — top-5 by similarity; (c) hybrid search — vector similarity AND metadata filter (category=Electronics AND price < 5000 AND in_stock=True); (d) update 10 documents and delete 5 — verify collection count. Implement the repository pattern: VectorStoreRepository abstract class with add(), search(), delete() methods; FAISSRepository and ChromaRepository implementations — swap with one line change. Use an AI assistant to explain when to use IVF vs. HNSW index — design 1 experiment to test the recommendation.

Unit IV: API Design and Microservice Integration:

10. FastAPI — Full REST API with Pydantic, Auth, Middleware, and Background Tasks

Build a FastAPI application exposing a machine learning prediction service. Endpoints: (a) POST /predict: accept {text: str, model_version: str}, validate with Pydantic (text min_length=1, max_length=1000, model_version pattern=r'v\d+'), run a mock classification (random label + confidence), return {label, confidence, latency_ms}; (b) GET /models: return list of available model versions from an in-memory dict; (c) POST /batch_predict: accept List[PredictRequest] (max 100), process in a background task using BackgroundTasks, return {job_id}; (d) GET /jobs/{job_id}: return job status and results when complete. Middleware: add RequestLoggingMiddleware (log method, path, status, latency to a JSON log file) and RateLimitMiddleware (max 10 requests/second per client IP using a sliding window counter in a dict). Security: implement OAuth2 password flow using fastapi.security.OAuth2PasswordBearer — protect /predict with a valid token check. Lifespan: load the "model" (a sklearn LogisticRegression trained on a toy dataset) on startup. Document all endpoints using FastAPI's built-in /docs. Use ChatGPT to design the Pydantic request and response models — review for missing validation rules and implement corrections.

11. gRPC Service and Circuit Breaker Pattern

Part A — gRPC: define a protobuf schema (prediction.proto) with: PredictRequest {text: string, model_id: string}, PredictResponse {label: string, confidence: float, latency_ms: int64}, and a service PredictionService with 3 RPCs: Predict (unary), PredictStream (server-streaming — stream progress updates during batch prediction), PredictBatch (client-streaming — send multiple texts, receive one aggregated response). Generate Python stubs using grpc_tools.protoc. Implement the server and a client. Benchmark: compare gRPC unary vs. REST (FastAPI) for 1000 identical prediction requests — measure latency (p50, p95, p99) and throughput (req/s). Part B — Circuit Breaker: implement a circuit breaker around a downstream service call (simulate failures by randomly raising exceptions 30% of the time): (a) use the circuitbreaker library with FAILURE_THRESHOLD=3, RECOVERY_TIMEOUT=10; (b) demonstrate the 3 states — Closed (normal), Open (fail-fast), Half-Open (probe) — by forcing failures and observing state transitions; (c) implement graceful degradation: when circuit is

Open, return a cached result instead of failing. Use Claude to explain the circuit breaker state machine — map each state to 1 of your experimental observations.

12. Async FastAPI, API Gateway Pattern, and Inter-Service Communication

Build 3 microservices: ServiceA (data-ingestion — accepts raw text), ServiceB (prediction — returns label + confidence), ServiceC (logging — stores request+response to SQLite). Each is a separate FastAPI app running on different ports. Async: implement async endpoints using async def and await; use aiohttp.ClientSession for HTTP calls between services (not requests); demonstrate that concurrent requests complete faster — send 50 concurrent requests using asyncio.gather() and compare with synchronous calls. API Gateway (mock): implement a 4th FastAPI service that acts as a gateway — receives a client request, authenticates the token, routes to ServiceA or ServiceB based on the path, applies rate limiting (10 req/s per IP), and logs to ServiceC via async call. Message-passing integration: ServiceB publishes prediction results to a Kafka topic predictions; ServiceC subscribes and persists. Demonstrate end-to-end flow: client → Gateway → ServiceA → ServiceB (Kafka publish) → ServiceC (Kafka consume → SQLite). Use an AI assistant to generate the aiohttp client session code for ServiceA-to-ServiceB communication — test and document corrections.

Unit V: Workflow Orchestration and Responsible Integration:

13. Apache Airflow — DAG Design, Branching, XCom, and Data Quality Gates

Install Airflow locally (pip install apache-airflow) and initialise the metadata database. Build a DAG ml_pipeline_dag (schedule='0 2 * * *', catchup=False) with: Task 1: PythonOperator — ingest_data: download a CSV from a URL, save to /tmp/raw.csv, push filename to XCom; Task 2: PythonOperator — validate_data: pull filename from XCom (ti.xcom_pull), check null rates, row count, schema; push validation_status (pass/fail); Task 3: BranchPythonOperator — branch_on_quality: read validation_status from XCom; if pass, go to process_data; if fail, go to alert_team; Task 4a: process_data — clean and transform, save to /tmp/processed.parquet; Task 4b: alert_team — BashOperator that echoes an alert message and exits; Task 5: ShortCircuitOperator — check_model_threshold: only continue if accuracy > 0.80 (read from XCom); Task 6: PythonOperator — register_model: simulate MLflow model registration. Configure retries=2, retry_delay=timedelta(minutes=1) on ingest_data and process_data. Embed a Great Expectations checkpoint inside validate_data: validate that null rate < 5% for all columns. Use ChatGPT to explain XCom serialisation limits — demonstrate the limit by attempting to push a large DataFrame and document the error and solution.

14. Pipeline Observability — Structured Logging, Prometheus Metrics, and OpenTelemetry Tracing

Instrument the FastAPI prediction service from Experiment 10 with full observability. Structured logging: configure Python logging with a JSON formatter (python-json-logger); log every request with: request_id (UUID), endpoint, method, status_code, latency_ms, model_version, client_ip; write to logs/app.jsonl; demonstrate log parsing using pandas.read_json(lines=True) and filtering for errors (status >= 400). Prometheus metrics: add prometheus-client to FastAPI; expose GET /metrics; define: (a) prediction_requests_total (Counter, labels: endpoint, status, model_version); (b) prediction_latency_seconds (Histogram, buckets=[0.01, 0.05, 0.1, 0.25, 0.5, 1.0]); (c) active_connections (Gauge). Send 200 requests with varied latency (random.uniform(0.01, 0.5)) and verify metrics at /metrics. OpenTelemetry: instrument with opentelemetry-sdk and opentelemetry-instrumentation-fastapi; create a trace for each request with 2 child spans: preprocess_input and run_model; export traces to OTLP (Jaeger

or console). Demonstrate distributed tracing: trace a request from the API Gateway (Exp 12) through to ServiceB, showing the trace propagated via HTTP headers. Use Claude to generate the JSON logging configuration — implement it and verify that all required fields appear in every log line.

15. Responsible Integration — PII Detection, Consent-Aware Routing, and GDPR/DPDP Compliance

PII detection pipeline: (a) build a PII detector using spaCy NER (en_core_web_sm) combined with regex patterns for email (RFC 5322), Indian mobile number (10 digits starting with 6–9), Aadhaar (12 digits with space every 4), and PAN (ABCDE1234F pattern); (b) load a synthetic dataset of 500 customer records containing PII; detect and log PII fields per record; (c) implement a PII redaction function: replace detected PII with [REDACTED-<TYPE>] before persistence; verify that redacted records contain no raw PII. Consent-aware data routing: (a) augment records with consent flags: {analytics_consent: bool, marketing_consent: bool, third_party_consent: bool}; (b) implement a DataRouter class with route(record) method: if analytics_consent=True, route to analytics_db (SQLite); if marketing_consent=True, route to marketing_db; if neither, route to minimal_storage_db (store only id, timestamp); (c) process all 500 records and verify each reached the correct destination. Right-to-erasure: implement delete_user(user_id) that: removes all records from all 3 SQLite databases where user_id matches; logs the deletion with timestamp and operator; verify with a post-deletion query. GDPR/DPDP checklist: produce a 15-item compliance checklist (purpose limitation, data minimisation, storage limitation, accuracy, security, right-to-access, right-to-erasure, consent withdrawal, cross-border transfer, DPO appointment, breach notification, DPDP Act 2023 data fiduciary obligations, children's data, automated decision-making, audit trail) — assess each as Implemented / Partial / Not Implemented for this experiment. Use an AI assistant to generate the DPDP Act 2023 obligations checklist — evaluate completeness against the official Act text (meity.gov.in) and document gaps.

AI Tools Integration (Lab):

ChatGPT / Claude for code generation, architecture design, pattern explanation, and checklist creation (all AI outputs independently verified and documented). kafka-python for Kafka producer-consumer. SQLAlchemy + Alembic for ORM and migrations. pymongo + redis-py for NoSQL integration. FAISS and Chroma for vector similarity search. sentence-transformers (HuggingFace) for embeddings. FastAPI + gRPC for API development. aiohttp for async HTTP. Apache Airflow (pip) for DAG orchestration. Great Expectations for inline data validation. Prometheus-client + OpenTelemetry for observability. spaCy (en_core_web_sm) for PII detection. MinIO + boto3 for S3-compatible storage. circuitbreaker for circuit breaker pattern.

Text Books:

1. Martin Kleppmann, Designing Data-Intensive Applications, O'Reilly Media, 2017.
2. Wes McKinney, Python for Data Analysis, 3rd Edition, O'Reilly Media, 2022.
3. Chip Huyen, Designing Machine Learning Systems, O'Reilly Media, 2022.
4. Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson.

Reference Books:

1. Sam Newman, Building Microservices, 2nd Edition, O'Reilly Media, 2021.
2. Bas Geerdink, A Practical Guide to Building Ethical AI Systems, Manning, 2023.

Online Learning Resources:

1. Apache Airflow – airflow.apache.org |FastAPI – fastapi.tiangolo.com |gRPC Python – grpc.io/docs/languages/python
2. SQLAlchemy – sqlalchemy.org | Apache Spark – spark.apache.org |kafka-python – kafka-python.readthedocs.io
3. FAISS – github.com/facebookresearch/faiss | Chroma – trychroma.com |HuggingFace – huggingface.co
4. Prometheus Python – prometheus.io/docs |OpenTelemetry Python – opentelemetry.io/docs/languages/python
5. India DPDP Act 2023 – meity.gov.in | GDPR – gdpr.eu |spaCy – spacy.io

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26PC05205P	PC	**Intelligent Database Management Systems Lab** (Common to CSE, AI&DS, CSE(AI) & CSE(A&ML))	0	0	0	3	1.5

Pre-Requisites: Knowledge of computer fundamentals, programming concepts, data structures, and basic mathematical foundations including sets and relations.

Course Objectives:

- To develop practical skills in database design and implementation through ER modelling, relational schema design, and database creation for real-world applications.
- To apply SQL techniques for database operations including data definition, manipulation, querying, optimization, indexing, and transaction control.
- To design efficient and reliable database systems using normalization, schema decomposition, concurrency control, and recovery mechanisms.
- To explore modern intelligent database technologies including NoSQL databases, graph databases, vector databases, and AI-enabled data management systems.
- To utilizer Artificial Intelligence tools and platforms to support database design, natural language querying, schema validation, query optimization, and intelligent analytics.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Design ER models and convert them into relational schemas for real-world applications.(L3)

CO2: Construct and execute SQL queries for data definition, manipulation, retrieval, and optimization.(L3)

CO3: Apply normalization techniques and schema decomposition methods to design efficient databases. (L3)

CO4: Implement transaction management, concurrency control, and recovery mechanisms in database environments.(L4)

CO5: Develop intelligent database applications using NoSQL systems, vector databases, and AI- assisted querying techniques. (L4)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	3	2	3		3	3	3		2	2	3
CO2	3	2	3	2	3		3	3	3		2	2	3
CO3	3	3	2	2	3		3	3	3		2	3	3
CO4	3	3	3	2	3		3	3	3		2	3	3
CO5	3	3	3	3	3		3	3	3		3	3	3

List of Experiments:

Module I: Database Modelling And SQL Foundations

AI Integration: Use draw.io and ChatGPT to generate and validate ER diagrams and schema designs.

Suggested Tools: draw.io; ChatGPT

Experiments:

Experiment 1: ER Modelling — Design an ER diagram for a real-world application (e.g., library, hospital, or e-commerce system) and convert it into a relational schema using ER-to-relational mapping rules.

Experiment 2: Database & DDL Operations — Create a database and perform DDL operations (CREATE, ALTER, DROP) with integrity constraints such as Primary Key, Foreign Key, CHECK, UNIQUE, and NOT NULL.

Experiment 3: DML & DQL Operations — Perform DML and DQL commands (INSERT, UPDATE, DELETE, SELECT) with filtering, sorting, grouping, and aggregate functions on the created schema.

Module II: SQL Programming And Query Optimization

AI Integration: Use GitHub Copilot and ChatGPT to generate SQL queries and compare execution plans.

Suggested Tools: GitHub Copilot; ChatGPT

Experiments:

Experiment 4: Joins & Subqueries — Develop SQL queries using joins, nested queries, subqueries, set operations, aggregate functions, GROUP BY, and HAVING clauses.

Experiment 5: Views, Indexes & Optimization — Create and analyze views and indexes, and perform query optimization comparing indexed and non-indexed access methods.

Experiment 6: Relational Algebra to SQL — Implement relational algebra operations and convert relational algebra expressions into equivalent SQL statements.

Module III: Database Design And Normalization

AI Integration: Use ChatGPT and Jupyter Notebook to assist normalization and validate using Armstrong's axioms.

Suggested Tools: ChatGPT; Jupyter Notebook

Experiments:

Experiment 7: Normalization (1NF–BCNF) — Identify functional dependencies and perform normalization of relations from 1NF through BCNF.

Experiment 8: Closures & Decomposition — Implement attribute closure computation, minimal cover generation, lossless decomposition, and dependency preservation.

Experiment 9: Schema Comparison — Compare normalized and denormalized schema designs for analytical workloads and document the trade-offs.

Module IV: Transaction Management And Recovery

AI Integration: Use Python simulations and ChatGPT to generate transaction schedules and precedence graph analysis.

Suggested Tools: Python; ChatGPT

Experiments:

Experiment 10: Transaction Control — Implement transaction control operations using COMMIT, ROLLBACK, and SAVEPOINT.

Experiment 11: Concurrency Control — Simulate concurrency control mechanisms including locking protocols, two-phase locking, deadlock detection, and serializability analysis.

Experiment 12: Logging & Recovery — Implement database recovery mechanisms including logging, checkpointing, undo-redo recovery, and Write Ahead Logging.

Module V: Intelligent Data Management For AI Applications

AI Integration: Use ChatGPT, LangChain, MongoDB Atlas, FAISS, Chroma, and Neo4j for intelligent database experimentation.

Suggested Tools: ChatGPT; LangChain; MongoDB Atlas; FAISS; Chroma; Neo4j

Experiments:

Experiment 13: NoSQL with MongoDB — Perform CRUD operations and aggregation queries using a NoSQL database such as MongoDB.

Experiment 14: Graph Databases with Neo4j — Develop a graph database application using Neo4j and implement Cypher queries.

Experiment 15: Intelligent Applications — Build intelligent database applications using Text-to-SQL, RAG-over-SQL, vector databases (FAISS/Chroma), similarity search, and AI-assisted data analytics.

Suggested Mini Projects:

- An AI-assisted Text-to-SQL query interface for a chosen database schema.
- A hybrid application combining a relational database (SQL) with a vector database (FAISS/Chroma) for semantic search.

Text Books:

1. Abraham Silberschatz, Henry Korth, S. Sudarshan, Database System Concepts, 7th Edition, McGraw-Hill.
2. Ramez Elmasri and Shamkant Navathe, Fundamentals of Database Systems, 7th Edition, Pearson.

Reference Books:

1. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill.
2. Martin Kleppmann, Designing Data-Intensive Applications, O'Reilly Media.

Online Learning Resources:

1. NPTEL – Database Management System – nptel.ac.in
2. MongoDB University – Free NoSQL Courses – university.mongodb.com
3. Neo4j GraphAcademy – Free Graph Database Courses – graphacademy.neo4j.com

Recommended Free & Open-Source AI / Computational Tools:

1. **AI Assistants:** ChatGPT; GitHub Copilot – query generation, debugging, and schema/normalization validation.
2. **Modelling Tools:** draw.io – ER diagram design.
3. **Databases & Engines:** MongoDB (Atlas free tier); Neo4j (Community/AuraDB Free) – NoSQL and graph database experimentation.
4. **Vector Search Libraries:** FAISS; Chroma – similarity search and vector database applications.
5. **Frameworks:** LangChain – building Text-to-SQL and RAG-over-SQL pipelines.
6. **Notebooks & Simulation:** Python & Jupyter Notebook – transaction simulations and normalization/closure computations.

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26SE30201S	SE	Data Analytics using Python (Common to CSE, AI&DS, CSE(AI) & CSE(A&ML))	1	0	0	2	2

Pre-Requisites: Basic knowledge of Python programming, data types, control structures, functions, and fundamental concepts of mathematics and statistics.

Course Objectives:

- To develop practical skills for handling, processing, and analyzing data using Python-based analytical tools and libraries.
- To enable students to perform data cleaning, transformation, aggregation, and exploratory analysis on structured datasets.
- To build competency in creating visualizations and dashboards for effective interpretation and communication of analytical insights.
- To introduce real-world analytics workflows involving data acquisition, processing, reporting, and presentation.
- To prepare students with a strong foundation in data analytics for advanced courses in Artificial Intelligence, Machine Learning, and Data Science.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Import, process, and manipulate datasets using Python analytical libraries. (L4)

CO2: Apply data cleaning, transformation, and aggregation techniques to prepare datasets for analysis. (L5)

CO3: Perform exploratory data analysis and create meaningful visualizations to derive insights. (L5)

CO4: Develop analytical reports and dashboards using Python tools and AI-assisted analytics workflows. (L6)

CO5: Design and implement data-driven analytical solutions using real-world datasets. (L6)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	2	2	3		3	3	3		1	2	3
CO2	3	3	2	2	3		3	3	3		1	3	3
CO3	3	3	2	3	3		3	3	3		1	3	3
CO4	3	3	3	3	3		3	3	3		1	3	3
CO5	3	3	3	3	3		3	3	3		1	3	3

List of Experiments:

Module I – Data Handling Using Python (Sessions 1–3)

Tools: Python 3, Jupyter Notebook, Pandas, Numpy

Topics: Data Analytics Workflow; Jupyter Notebook Environment; Data Sources And File Formats; Importing Data; Csv, Excel And Json Processing; Numpy Arrays; Data Manipulation; Vectorised Operations; Data Export.

1. Dataset Import and Exploration

Import datasets from CSV (pd.read_csv), Excel (pd.read_excel), and JSON (pd.read_json) formats into Pandas DataFrames. Examine dataset dimensions (shape), column names (columns), data types (dtypes), and memory usage (info()). Compute summary statistics (describe()) for numerical columns; count missing values per column (isnull().sum()); display the first and last 5 rows (head(), tail()). Export the cleaned dataset to a new CSV and JSON file using to_csv() and to_json(). Use ChatGPT or NotebookLM to explain any unfamiliar statistics in the describe() output; document each explanation.

2. Numerical Data Processing using NumPy

Create 1D and 2D NumPy arrays from Python lists and using np.arange(), np.linspace(), np.zeros(), np.ones(), np.eye(). Perform indexing (arr[i]), slicing (arr[1:4]), and fancy indexing (arr[[0,2,4]]). Apply aggregation functions: np.sum(), np.mean(), np.std(), np.min(), np.max(), np.median() on both axes. Implement vectorised operations: element-wise arithmetic, broadcasting a scalar across an array, and computing the dot product of two 2D arrays. Compare execution time of a NumPy vectorised operation vs. an equivalent Python loop for array of 1 million elements using time module.

3. DataFrame Operations using Pandas

Create a DataFrame from a dictionary with 6 columns and 20 rows of student data (roll_no, name, branch, marks_list, attendance, grade). Perform: column selection (single and multiple), row filtering using boolean conditions (marks > 60 and attendance >= 75), loc[] and iloc[] indexing, and sorting by marks descending. Compute group-level aggregations using groupby('branch').agg({'marks': ['mean','max','min'], 'attendance': 'mean'}); display results in a formatted table. Add a derived column pass_fail (Pass if marks >= 50, Fail otherwise) using np.where(); export the final DataFrame to Excel. Use Claude or ChatGPT to explain the difference between loc[] and iloc[]; verify the explanation using 3 examples.

Module II – Data Processing and Transformation using Pandas (Sessions 4–6)

Tools: Pandas, Jupyter Notebook

Topics Series and DataFrames; Data Selection; Indexing and Filtering; Missing Data Handling; Data Cleaning; Aggregation; Grouping; Merging; Sorting; Feature Construction.

4. Data Cleaning and Missing Value Handling

Load the Titanic dataset (or any dataset with missing values and duplicates). Identify: duplicate rows (duplicated().sum()); missing values per column (isnull().sum(), visualised using a missingno bar chart); outliers in the fare/age column using Z-score ($|z| > 3$) and IQR ($1.5 \times IQR$) methods. Remove duplicate rows; impute missing Age with median, Embarked with mode; drop the Cabin column (>70% missing) and justify the decision. Apply Winsorisation to the Fare column (clip at 1st and 99th percentile); compare distribution before and after using histograms. Use ChatGPT to recommend the appropriate missing data strategy for each column; evaluate the suggestion and document whether you accepted or modified it.

5. Data Transformation and Feature Construction

Using the Ames Housing or NYC Taxi dataset: apply melt() to convert a wide-format DataFrame to long format; verify row count; apply pivot_table() to reconstruct the wide format and compare. Merge two related DataFrames (e.g., customers and orders) using inner, left, right, and outer joins — report row counts for each join type. Create 3 derived columns: price_per_sqft (price/area), age_of_house (current_year − year_built), is_recent (age_of_house ≤ 10). Apply groupby with transform() to add group-mean price as a new column without reducing the DataFrame. Use an AI

assistant to suggest 2 additional feature ideas; implement both and validate that they are non-leaking (not derived from the target variable).

6. Descriptive Statistics and Data Analysis
Load any domain dataset (Iris, Wine Quality, or Student Performance). Compute: measures of central tendency — mean, median, mode using Pandas and scipy.stats; measures of dispersion — variance, standard deviation, range, IQR; shape statistics — skewness and kurtosis using df.skew() and df.kurtosis(). Generate a cross-tabulation of two categorical variables using pd.crosstab() with normalize=True; interpret the conditional frequency distribution. Identify columns with |skewness| > 1; apply log1p transformation to reduce skewness; compare before and after using describe(). Use NotebookLM — upload the dataset documentation as a source and ask it to explain each statistical measure in context; document the explanation.

Module III – Exploratory Data Analysis and Visualization (Sessions 7–10)
Tools: Matplotlib, Seaborn, Plotly
Topics Descriptive Statistics; Data Exploration; Correlation Analysis; Visualization Principles; Line Charts; Bar Charts; Scatter Plots; Histograms; Boxplots; Heatmaps; Interactive Visualization.

7. Data Visualization using Matplotlib
Using a sales or student dataset, create 4 chart types with Matplotlib: (a) line chart — monthly revenue trend with markers and a 3-month rolling average overlay; (b) bar chart — top 10 products by sales volume (horizontal, colour-coded by category); (c) histogram — distribution of marks with 20 bins and a KDE curve overlaid using scipy.stats.gaussian_kde; (d) pie chart — market share by product category. Apply chart design principles to each: proper axis labels, title, legend, colour palette (avoid red-green for accessibility), and grid lines. Use ChatGPT to recommend a chart type for 3 new analysis questions; implement the recommended charts and document whether each recommendation was appropriate.

8. Advanced Visualization using Seaborn
Using the Titanic or Tips dataset, create 5 Seaborn plots: (a) scatter plot with regression line (sns.regplot) — age vs. fare, coloured by survived; (b) boxplot (sns.boxplot) — fare distribution per passenger class; (c) violin plot (sns.violinplot) — age distribution per sex and survived; (d) heatmap (sns.heatmap) — correlation matrix of all numerical columns with annotations and a diverging colour palette; (e) pair plot (sns.pairplot) — all numerical columns coloured by survival status. For each plot, write a 2-sentence interpretation of the insight shown. Use an AI assistant to generate an alternate seaborn palette recommendation and test it on 2 charts.

9. Interactive Visualization and Dashboard Design
Using Plotly Express and Plotly Graph Objects, build 3 interactive charts on a real dataset: (a) interactive line chart (px.line) with hover tooltips showing exact values and a range slider for the x-axis; (b) interactive scatter plot (px.scatter) with size mapped to a third variable, colour mapped to a category, and a dropdown filter for year; (c) animated bar chart (px.bar with animation_frame) showing top-5 values changing over time. Combine all 3 charts into a Plotly subplot (make_subplots) dashboard. Export the dashboard as an HTML file viewable in any browser. Use ChatGPT to suggest the best Plotly chart type for 2 additional analysis questions; justify your implementation choice.

10. Correlation and Comparative Data Analysis

Using a multivariate dataset (Wine Quality, Boston Housing, or Student Performance): compute pairwise Pearson, Spearman, and Kendall correlation coefficients for all numerical columns; display as three separate annotated heatmaps and identify the top-3 and bottom-3 correlated feature pairs. Perform comparative analysis: compare mean marks across 3 student groups using a grouped bar chart with error bars (±1 std); apply a t-test (scipy.stats.ttest_ind) to verify if the difference between two groups is statistically significant ($p < 0.05$). Identify 3 pairs of variables with strong correlation ($|r| > 0.7$); create scatter plots with regression lines for each. Document whether correlation implies causation for each pair; use Claude to verify your reasoning.

Module IV – Data Analytics Using Real-World Datasets (Sessions 11–13)

Tools: Pandas, Requests, Plotly

Topics Data Collection Methods; Public Dataset Usage; JSON Processing; API-based Data Collection; Time-Series Data Handling; Data Transformation; Report Generation; Dashboard Concepts.

11. Data Collection using APIs and JSON Processing

Collect data from two public REST APIs using the requests library: (a) Open-Meteo API (free, no key) — fetch 30 days of hourly temperature, humidity, and wind speed for two cities; load JSON response into a DataFrame using pd.json_normalize(); compute daily aggregates (min, max, mean); plot a dual-city comparison line chart; (b) REST Countries API (free, no key) — fetch all countries; extract name, capital, population, area, and region; load into a DataFrame; find the top 10 most populous countries per region. Handle API errors (timeout, 404, invalid city) with try-except; document all error cases and the handling applied.

12. Time-Series and Trend Analysis

Load a time-series dataset (daily stock prices from Yahoo Finance using yfinance, or provided weather/sales CSV). Parse the date column using pd.to_datetime() and set as DatetimeIndex. Perform: resampling — daily data to weekly mean and monthly sum using resample(); rolling statistics — 7-day and 30-day rolling mean using rolling().mean(); EWM — exponential weighted mean (ewm(span=7).mean()) to smooth short-term fluctuations. Decompose the time series into trend, seasonality, and residual using statsmodels.tsa.seasonal_decompose(). Plot all components on a 4-panel Matplotlib figure. Identify and annotate 3 significant events (peaks/troughs) on the trend line. Use ChatGPT to interpret the seasonality component; document the explanation.

13. Analytical Report Generation

Using the complete dataset from any previous experiment, generate a structured analytical report as a Jupyter Notebook: (a) Executive Summary — 5-sentence description of dataset, analysis goals, and key findings; (b) Dataset Overview — shape, column descriptions, data types, and quality summary (null rates, duplicate count, outlier count); (c) Key Metrics — top-5 summary statistics, distribution of target variable; (d) Insights — 5 numbered insights each supported by one chart; (e) Recommendations — 3 data-driven recommendations for a business decision-maker. Export the notebook as a PDF using nbconvert. Use NotebookLM — upload the notebook as a source; generate an audio overview; share the audio summary link and include the transcript as an appendix to the report.

Module V – Applied Data Analytics Project (Sessions 14–15)

Tools: Python, Jupyter Notebook, Plotly

Topics :Analytics Workflow; Data Preparation; Data Visualization; Insight Generation; Dashboard Development; Documentation; Project Presentation.

14. Dashboard Development using Real-World Datasets

Select a domain dataset (Education — student performance, Healthcare — patient records, Retail — sales data, or Weather — daily weather data). Build a complete multi-panel Plotly dashboard (minimum 6 charts) covering: dataset overview (bar/pie chart of category distribution), trend analysis (line chart with rolling average), comparative analysis (grouped bar with error bars), correlation analysis (annotated heatmap), outlier identification (boxplot), and a KPI summary panel (4 metric cards showing totals, averages, max/min). Add a dropdown filter (by category, year, or region) that updates all charts simultaneously. Export as an HTML file and a PDF report. Use an AI tool to suggest 2 additional KPIs relevant to the chosen domain; implement both.

15. Applied Data Analytics Mini Project

Design and implement a complete end-to-end data analytics solution (choose from: Student Performance Analytics, Retail Sales Analytics, Healthcare Analytics, Weather Analytics, Financial Data Analytics, or Social Media Analytics). The project must include: (a) data collection from at least 2 sources (CSV + API or two CSV files); (b) data cleaning report (missing values, duplicates, outliers — before and after statistics); (c) exploratory analysis with minimum 8 visualisations across at least 4 chart types; (d) a Plotly interactive dashboard with dropdown filter; (e) a time-series or correlation analysis; (f) a written insights report (minimum 5 data-driven insights with supporting charts); (g) an AI usage log documenting every AI tool used, the prompts given, and whether the output was accepted, modified, or rejected. Present in a 10-minute demo with Q&A.

Text Books:

1. Wes McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy and Jupyter, 3rd Edition, O'Reilly Media.
2. Jake VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media. (Free at jakevdp.github.io/PythonDataScienceHandbook)
3. Joel Grus, Data Science from Scratch: First Principles with Python, 2nd Edition, O'Reilly Media.

Reference Books:

1. Cathy O'Neil and Rachel Schutt, Doing Data Science, O'Reilly Media.
2. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow, 3rd Edition, O'Reilly Media. (Selected chapters on data preparation.)
3. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, 2nd Edition, O'Reilly Media. (Free at greenteapress.com)

Online Learning Resources:

1. Pandas Documentation – pandas.pydata.org | NumPy Documentation – numpy.org
2. Matplotlib Documentation – matplotlib.org | Seaborn Documentation – seaborn.pydata.org
3. Plotly Documentation – plotly.com/python |Jupyter Documentation – jupyter.org/documentation
4. Kaggle (free datasets and notebooks) – kaggle.com | Google Colab – colab.research.google.com
5. Python Official Documentation – docs.python.org/3 |FreeCodeCamp Data Analysis – freecodecamp.org
6. NotebookLM – notebooklm.google.com | NPTEL – Data Science for Engineers – nptel.ac.in

B. Tech. – II Year II Semester

Course Code	Category	Name of the Course	L	T	Pr	P	C
26BSHB214A	MC	Environmental Science (Common to all branches of Engineering)	2	0	0	0	0

Pre-Requisites: Basic understanding of science, environmental concepts, and computer fundamentals with an interest in sustainability, ecology, and societal issues related to the environment.

Course Objectives:

- To create awareness about environmental issues, natural resource management and the need for sustainable development.
- To explain the structure and function of various ecosystems and the significance of biodiversity and its conservation.
- To describe causes, effects and control of various types of pollution and principles of solid waste management.
- To examine social, ethical and legislative issues related to environment and sustainable development in India.
- To understand human population dynamics, environmental health linkages and the role of IT and AI tools in environmental monitoring.

Course Outcomes (COs):

On successful completion of the course, Student will be able to

CO1: Analyze the multidisciplinary nature of environmental studies and evaluate the impact of the use, over-exploitation, and management of forest, water, mineral, food, and energy resources through Indian and global case studies. (L4)

CO2: Analyze the structure and functioning of ecosystems, ecological succession, food chains, ecological pyramids, and biodiversity patterns, and assess the significance of biodiversity conservation strategies at local, national, and global levels. (L6)

CO3: Evaluate the causes, impacts, and control measures of various forms of environmental pollution, and assess the effectiveness of solid waste management and disaster management practices in different environmental contexts. (L5)

CO4: Evaluate the causes, impacts, and control measures of various forms of environmental pollution, and assess the effectiveness of solid waste management and disaster management practices in different environmental contexts. (L5)

CO5: Analyze population–environment interactions, environmental health issues, and the role of information technology and AI-based tools in environmental monitoring, public health surveillance, field data collection, and environmental documentation (L5)

CO – PO Articulation Matrix:

Scale: 3 – Strong | 2 – Moderate | 1 – Slight

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	2	1			1	3	2				2	2	2
CO2	2	2		1	2	3	2				2	2	2
CO3	2	2	2	2	2	3	3				2	1	2
CO4	2	2	2	2	2	3	3				3	2	2
CO5	1	1		1	3	3	2				3	2	2

List of Experiments:

Unit I: Multidisciplinary Nature of Environmental Studies and Natural Resources:

Multidisciplinary Nature of Environmental Studies: – Definition, Scope and Importance – Need for Public Awareness.

Natural Resources: Renewable and non-renewable resources – Natural resources and associated problems – Forest resources – Use and over-exploitation, deforestation, case studies – Timber extraction – Mining, dams and other effects on forest and tribal people – Water resources – Use and over utilization of surface and ground water – Floods, drought, conflicts over water, dams – benefits and problems – Mineral resources: Use and exploitation, environmental effects of extracting and using mineral resources, case studies – Food resources: World food problems, changes caused by agriculture and overgrazing, effects of modern agriculture, fertilizer-pesticide problems, water logging, salinity, case studies – Energy resources: Use and exploitation of conventional and non-conventional energy sources, case studies.

AI Integration: AI in Natural Resource Monitoring: Remote sensing and satellite imagery analysis using Google Earth Engine (free cloud platform) for real-time deforestation detection and land use change mapping. Machine learning for groundwater level prediction from historical well data (Python/scikit-learn). AI-based precision agriculture – reducing fertilizer and pesticide use through crop health monitoring using drone imagery and computer vision (overview). ML forecasting of renewable energy output (solar irradiance, wind speed) for energy resource planning.
Free / Open-Source AI Tools: Google Earth Engine (free cloud GIS); QGIS (free, open-source); Python (scikit-learn, Matplotlib) – Google Colab (free); Global Forest Watch (free dashboard).

Unit II: Ecosystems and Biodiversity and its Conservation:

Ecosystems: Concept of an ecosystem – Structure and function of an ecosystem – Producers, consumers and decomposers – Energy flow in the ecosystem – Ecological succession – Food chains, food webs and ecological pyramids – Introduction, types, characteristic features, structure and function of: Forest ecosystem; Grassland ecosystem; Desert ecosystem; Aquatic ecosystems (ponds, streams, lakes, rivers, oceans, estuaries).

Biodiversity and its Conservation: Introduction – Definition: genetic, species and ecosystem diversity – Bio-geographical classification of India – Value of biodiversity: consumptive use, productive use, social, ethical, aesthetic and option values – Biodiversity at global, national and local levels – India as a mega-diversity nation – Hot-spots of biodiversity – Threats to biodiversity: habitat loss, poaching of wildlife, man-wildlife conflicts – Endangered and endemic species of India – Conservation of biodiversity: In-situ and Ex-situ conservation of biodiversity.

AI Integration: AI in Ecosystem and Biodiversity Studies: AI-powered species identification – applications such as iNaturalist (free) use image recognition to identify plants, animals and insects from photographs, enabling citizen science biodiversity surveys. ML for biodiversity hotspot mapping using satellite imagery and species distribution models (MaxEnt – free). Computer vision for automated wildlife camera trap image analysis – identifying and counting species without manual review. Google Earth Engine for ecosystem health monitoring: NDVI time-series analysis for vegetation cover change detection. AI-assisted assessment of in-situ vs. ex-situ conservation effectiveness.
Free / Open-Source AI Tools: iNaturalist (free app/web); MaxEnt (free species distribution model); Google Earth Engine (free); QGIS (free); Python (rasterio, geopandas) – Google Colab.

Unit III: Environmental Pollution and Solid Waste Management:

Environmental Pollution: Definition, cause, effects and control measures of: Air Pollution; Water Pollution; Soil Pollution; Marine Pollution; Noise Pollution; Thermal Pollution; Nuclear Hazards.

Solid Waste Management: Causes, effects and control measures of urban and industrial wastes – Role of an individual in prevention of pollution – Pollution case studies – Disaster Management: floods, earthquake, cyclone and landslides.

AI Integration: AI in Pollution Monitoring and Disaster Management: ML-based Air Quality Index (AQI) prediction from meteorological data and traffic density (Python/scikit-learn/TensorFlow – free). IoT sensor networks with AI-driven anomaly detection for real-time water quality monitoring. Computer vision for automated solid waste sorting and classification (overview of AI-enabled sorting robots). OpenAQ platform (free, open data) for accessing global air quality sensor data for ML model training. AI and GIS (QGIS) for disaster risk zone mapping – flood inundation modelling, earthquake vulnerability assessment and cyclone track prediction.
Free / Open-Source AI Tools: OpenAQ (free global AQ data API); QGIS (free); Python (scikit-learn, TensorFlow, Matplotlib) – Google Colab (free); CPCB data portal (free).

Unit IV: Social Issues and the Environment:

Social Issues and the Environment: From unsustainable to sustainable development – Urban problems related to energy – Water conservation, rain water harvesting, watershed management – Resettlement and rehabilitation of people; its problems and concerns, case studies – Environmental ethics: issues and possible solutions – Climate change, global warming, acid rain, ozone layer depletion, nuclear accidents and holocaust, case studies – Wasteland reclamation – Consumerism and waste products – Environment Protection Act – Air (Prevention and Control of Pollution) Act – Water (Prevention and Control of Pollution) Act – Wildlife Protection Act – Forest Conservation Act – Issues involved in enforcement of environmental legislation – Public awareness.

AI Integration: AI for Climate Change and Sustainable Development: Global climate model data analysis using Python (xarray, Matplotlib) from freely available CMIP6 datasets – visualising temperature rise, precipitation change and sea level scenarios. ML for carbon footprint estimation from energy consumption and transport data (Python/scikit-learn). AI-based watershed management: QGIS + Python for delineation, runoff modelling and rainwater harvesting potential mapping. Overview of AI in environmental compliance monitoring – automated identification of violations using satellite data. Digital tools for public environmental awareness: social media sentiment analysis for pollution perception studies (Python/NLP).

Unit V: Human Population and the Environment:

Human Population and the Environment: Population growth, variation among nations – Population explosion – Family Welfare Programmes – Environment and human health – Human Rights – Value Education – HIV/AIDS – Women and Child Welfare – Role of Information Technology in Environment and human health – Case studies.

Field Work: Visit to a local area to document environmental assets: river / forest / grassland / hill / mountain – Visit to a local polluted site: Urban / Rural / Industrial / Agricultural – Study of common plants, insects and birds – river, hill slopes, etc.

AI Integration:

AI in Population Studies and Public Health: ML-based population growth forecasting from census data using Python (time-series regression with scikit-learn / LSTM with TensorFlow). AI in public health surveillance – disease outbreak prediction and mapping using spatiotemporal ML

models (overview of WHO FluNet, IDSP data). Role of Information Technology and AI in environmental health: wearable sensors for personal air quality monitoring; smartphone-based water quality testing with ML image analysis. Citizen science apps for field work data collection: iNaturalist, eBird (free) for recording plants, birds and insects during field visits; OpenStreetMap for documenting environmental assets. GIS-based environmental health mapping using QGIS (free).

Text Books:

1. Erach Bharucha, "Textbook of Environmental Studies for Undergraduate Courses", for University Grants Commission, Universities Press.
2. Palaniswamy, "Environmental Studies", Pearson Education.
3. S. Azeem Unnisa, "Environmental Studies", Academic Publishing Company.
4. K. Raghavan Nambiar, "Text Book of Environmental Studies for Undergraduate Courses as per UGC Model Syllabus", Scitech Publications (India) Pvt. Ltd.

Reference Books:

1. Deeksha Dave and E. Sai Baba Reddy, "Textbook of Environmental Science", Cengage Publications.
2. M. Anji Reddy, "Text Book of Environmental Sciences and Technology", BS Publications.
3. J. P. Sharma, "Comprehensive Environmental Studies", Laxmi Publications.
4. J. Glynn Henry and Gary W. Heinke, "Environmental Sciences and Engineering", Prentice Hall of India Private Limited.
5. G. R. Chatwal, "A Text Book of Environmental Studies", Himalaya Publishing House.
6. Gilbert M. Masters and Wendell P. Ela, "Introduction to Environmental Engineering and Science", Prentice Hall of India Private Limited.

Online Learning Resources:

1. https://www.coursera.org/learn/python-for-applied-data-science-ai
2. https://www.coursera.org/learn/python?specialization=python#syllabus
3. NPTEL: Environmental Science – https://nptel.ac.in/courses/124107002
4. Google Earth Engine (free cloud geospatial platform) – https://earthengine.google.com
5. Global Forest Watch (free deforestation monitoring) – https://www.globalforestwatch.org
6. OpenAQ (free global air quality open data) – https://openaq.org
7. iNaturalist (free citizen science biodiversity app) – https://www.inaturalist.org
8. QGIS (free, open-source GIS) – https://qgis.org
9. India Meteorological Department (free data) – https://mausam.imd.gov.in
10. Central Pollution Control Board (CPCB) data portal (free) – https://cpcb.nic.in