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

## DEPARTMENT OF
## COMPUTER SCIENCE AND ENGINEERING

# 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**

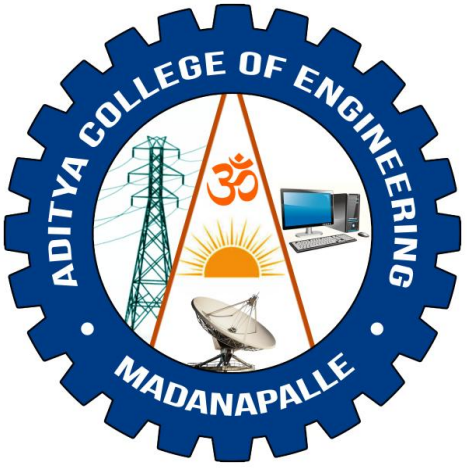
# ADITYA COLLEGE OF ENGINEERING, MADANAPALLE

**(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

# COMPUTER SCIENCE AND ENGINEERING

# 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 Computer Science and Engineering

**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 develop the students into globally competent professionals in the domain of Computer science and Engineering to solve real-world problems with professional ethics and committed social values.

**Department Mission:**

- To provide the ambiance to make the students industry - ready, Researchers and Entrepreneurs by offering courses on cutting - edge technologies.
- To Impart high-quality experiential learning to get expertise in modern software tools and to cater to the real-time requirements of the industry.
- To inculcate the qualities of professional ethics and a sense of social responsibility among the students.

**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 (Computer Science and Engineering) will be able to,

- **PSO1:** Ability to understand and apply the concepts of mathematics, basic sciences, basic computing, in analyzing the real world problems and solving them.
- **PSO2:** Ability to design and develop computer programs / computer - based systems in the areas related to algorithms, networking, web design, and data analytics using Software engineering principles and practices.

**Programme Educational Objectives (PEOs):**

After few years of graduation, the graduates of Computer Science and Engineering 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. – Computer Science and Engineering
# 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 - A

## 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 | 26BSHB106T | Physics for Advanced Computing | 2 | 1 | 0 | 0 | 3 |
| 3. | ES | 26ES03101T | Engineering Graphics | 1 | 0 | 4 | 0 | 3 |
| 4. | ES | 26ES05101T | Computational Thinking & Problem Solving with Python | 3 | 0 | 0 | 0 | 3 |
| 5. | ES | 26ES31101T | AI Tools and Applications | 1 | 0 | 4 | 0 | 3 |
| 6. | BS | 26BSHB106P | Physics for Advanced Computing Lab | 0 | 0 | 0 | 3 | 1.5 |
| 7. | ES | 26ES05101P | Computational Thinking & Problem Solving with Python Lab | 0 | 0 | 0 | 3 | 1.5 |
| 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** | **10** | **21** |

## 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 | 26BSHB110T | Applied Chemistry for Computer Science | 3 | 0 | 0 | 0 | 3 |
| 3. | HM | 26HMHS101T | Communicative English | 2 | 0 | 0 | 0 | 2 |
| 4. | ES | 26ES03103T | Design Thinking | 1 | 0 | 4 | 0 | 3 |
| 5. | PC | 26PC05101T | Data Structures & Efficient Problem Solving | 2 | 0 | 2 | 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. | PC | 26PC05101P | Data Structures & Efficient Problem-Solving Lab | 0 | 0 | 0 | 2 | 1 |
| 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** | **11+2** | **0** | **8** | **8** | **19** |

## B. Tech. – II Year I 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 | 26ES01201T | Sustainability and Green Engineering (AI Integrated) | 2 | 0 | 0 | 0 | 2 |
| 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 | 26PC04203P | 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 | 26BSHB214A | Environmental Science | 2 | 0 | 0 | 0 | 0 |
| | | | **Total** | **16+2** | **0** | **2** | **10** | **22** |

## B. Tech. – II Year II 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 | 26PC05204T | Operating Systems and Intelligent Resource Management | 3 | 0 | 0 | 0 | 3 |
| 4. | PC | 26PC05205T | Intelligent Database Management Systems | 3 | 0 | 0 | 0 | 3 |
| 5. | PC | 26PC05206T | Automata Theory and Compiler Design | 2 | 0 | 2 | 0 | 3 |
| 6. | PC | 26PC05207T | Software Engineering for AI Driven Systems | 2 | 0 | 2 | 0 | 3 |
| 7. | PC | 26PC05204P | Operating Systems and Intelligent Resource Management 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 | 26HMHS206A | Technical Paper Writing and IPR | 2 | 0 | 0 | 0 | 0 |
| | | | **Total** | **15+2** | **0** | **4** | **10** | **22** |
| | 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 |
|---|---|---|---|---|---|---|---|
| 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 I 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 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 |
|---|---|---|---|---|---|---|---|
| 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 I 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)
2. TensorFlow / Scikit-learn

**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 ($\pi r^2$) 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 |
|---|---|---|---|---|---|---|---|
| 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 I 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. – 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 | 2 | 1 |
| CO2 | 3 | 3 | 3 | 2 | 2 | | 1 | | | | 2 | 2 | 1 |
| CO3 | 3 | 2 | 2 | 2 | 1 | | 1 | | | | 2 | 2 | 1 |
| CO4 | 3 | 3 | 3 | 3 | 2 | | 1 | | | | 2 | 2 | 1 |
| CO5 | 3 | 3 | 2 | 3 | 3 | | 1 | | | | 2 | 2 | 1 |

**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 ($\nabla$) 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:**

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

## B. Tech. – I Year II 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 $\Psi$ and $\Psi^2$. 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. $\pi$-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 II 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 II 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 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 (inorder, preorder, postorder, 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 |
|---|---|---|---|---|---|---|---|
| 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 II 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 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 |
|---|---|---|---|---|---|---|---|
| 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 II 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. – II Year
# I Semester

## B.Tech. – II Year I 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 | 1 |
| CO2 | 3 | 2 | | 1 | | | 1 | | | | 1 | 2 | 1 |
| CO3 | 3 | 2 | 1 | 1 | 1 | | 1 | | | | 1 | 2 | 1 |
| CO4 | 3 | 3 | 1 | 2 | 1 | | 1 | | | | 1 | 2 | 1 |
| CO5 | 3 | 3 | 2 | 3 | 1 | | 1 | | | | 1 | 2 | 2 |

**Unit I: Descriptive Statistics:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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 Resources:**

- NPTEL: Probability and Statistics – https://nptel.ac.in
- MIT OpenCourseWare – Introduction to Probability and Statistics: https://ocw.mit.edu
- Python SciPy Statistics Documentation: https://docs.scipy.org/doc/scipy/reference/stats.html
- Statsmodels Documentation: https://www.statsmodels.org
- Google Colab for Python: https://colab.research.google.com

**Web Resources & NPTEL Links:**

- NPTEL: Probability and Statistics -- https://nptel.ac.in
- MIT OpenCourseWare -- Introduction to Probability and Statistics: https://ocw.mit.edu
- Python SciPy Statistics Documentation: https://docs.scipy.org/doc/scipy/reference/stats.html
- Google Colab for Python: https://colab.research.google.com
- NumPy Documentation: https://numpy.org/doc
- Statsmodels Documentation: https://www.statsmodels.org
- WolframAlpha (free tier) for verification: https://www.wolframalpha.com

**Recommended Free & Open-Source AI/Computational Tools:**

- Python (NumPy, SciPy, Matplotlib, Pandas) -- FREE on Google Colab for statistical computation, distribution analysis, and visualization.
- SymPy (free, open-source) -- For symbolic probability derivations, distribution functions, and algebraic verification.
- Seaborn (free, open-source) -- For statistical data visualization including correlation heatmaps, distribution plots, and regression plots.
- scikit-learn (free, open-source) -- For regression modelling, correlation analysis, and exploratory data analysis.
- Jupyter Notebook (free, open-source) -- For interactive statistical computation and lab documentation.
- statsmodels (free, open-source) -- For advanced hypothesis testing, regression analysis, and time series statistics.
- GNU Octave (free, open-source) -- MATLAB-like numerical computing for statistical and probability workflows..

**B.Tech. – II Year I 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 understanding of mathematics and elementary concepts of business and commerce. Familiarity with basic concepts of accounting, economics, and financial transactions.

**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 concepts of economics including demand, supply, market structures, and the time value of money to analyze 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:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core Content:** Accounting Principles and Digital Accounting Systems; Journal, Ledger, Trial Balance; Final Accounts: Trading Account, Profit and Loss Account, Balance Sheet with Simple Adjustments.

Financial Analysis: Liquidity Ratios, Activity Ratios, Capital Structure Ratios, Profitability Ratios.

**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 Resources:**

- NPTEL (IIT): Managerial Economics – https://nptel.ac.in/courses/110105063
- NPTEL (IIT): Financial Accounting for Engineers – https://nptel.ac.in/courses/110107114
- Coursera: Economics for Non-Economists – https://www.coursera.org/learn/economics
- Khan Academy: Microeconomics & Finance Basics
- SWAYAM Portal: Business Economics – https://swayam.gov.in

| S.No. | Platform | Course / Resource | Link / Access |
|---|---|---|---|
| 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 (Univ. of California) | https://www.coursera.org/learn/economics |
| 4 | edX | Financial Accounting (MIT OpenCourseWare) | https://www.edx.org/course/financial-accounting |
| 5 | Khan Academy | Microeconomics & Finance Basics | https://www.khanacademy.org/economics-finance-domain |
| 6 | SWAYAM Portal | Business Economics (UGC-approved) | https://swayam.gov.in |

**AI TOOLS USED IN THIS COURSE:**

| S.No. | AI Tool | Application in Course |
|---|---|---|
| 1 | ChatGPT / Claude AI | Concept clarification, case study generation, problem solving assistance |
| 2 | Microsoft Copilot | Financial analysis, spreadsheet automation, data interpretation |
| 3 | Google Gemini | Research support, market structure comparisons, summarisation |
| 4 | Tally AI / Zoho Books AI | Digital accounting simulations, ledger preparation, journal entries |
| 5 | Statista / Tableau AI | Data visualisation for demand trends and financial ratios |
| 6 | Wolfram Alpha | Quantitative problems – NPV, IRR, Break-Even, elasticity calculations |
| 7 | Grammarly AI | Report writing, case study documentation, assignments |

**AI-INTEGRATED ACTIVITIES:**

| Unit | Activity | Description | AI Tool Used | Outcome Mapped |
|---|---|---|---|---|
| I | Demand Forecasting Simulation | Use AI tools to forecast demand for a consumer product using historical data | ChatGPT + Statista | CO1 |
| I | Elasticity Chatbot Quiz | AI-generated adaptive quiz on types and measurement of elasticity | Google Gemini | CO1 |
| II | Break-Even Calculator | Build a Break-Even model using Excel + Copilot for sensitivity analysis | Microsoft Copilot | CO2 |
| II | Cost Curve Visualisation | Plot and interpret cost curves using AI-assisted data tools | Wolfram Alpha / Tableau | CO2 |
| III | Market Structure Case Study | Generate real-world case studies of oligopoly (e.g., Telecom sector) using AI | Claude AI / ChatGPT | CO3 |
| III | Startup Evaluation Report | Evaluate a digital startup's business model using AI-generated insights | Google Gemini | CO3 |
| IV | Capital Budgeting Decision Tool | Use AI to compare NPV, IRR for two competing engineering projects | Copilot + Wolfram Alpha | CO4 |
| IV | IRR Sensitivity Analysis | Explore how changes in cash flows affect IRR using AI-powered spreadsheets | Microsoft Copilot | CO4 |
| V | Digital Accounting Exercise | Prepare journal, ledger, and trial balance using Tally AI / Zoho Books | Tally AI / Zoho AI | CO5 |
| V | Financial Ratio Dashboard | Construct a financial ratio dashboard from a company's annual report using Tableau AI | Tableau AI | CO5 |

**B.Tech. – II Year I 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 understanding of environmental science and engineering concepts. Familiarity with basic mathematics, energy concepts, and computer 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, SDGs, the carbon cycle, and greenhouse gas accounting. (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 | 2 | 1 |
| CO2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | | | | 2 | 2 | 1 |
| CO3 | 2 | 2 | 2 | 2 | 3 | 3 | 1 | | | | 2 | 2 | 2 |
| CO4 | 2 | 2 | 3 | 2 | 3 | 3 | 2 | | | | 3 | 2 | 2 |
| CO5 | 2 | 3 | 3 | 3 | 3 | 3 | 2 | | | | 3 | 2 | 2 |

**Unit I: Fundamentals of Sustainability and Carbon Cycle:**

**Core 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:**
**Core 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:**
**Core 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:**
**Core 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:**
**Core 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 Resources:**

- NPTEL: Sustainable Development – https://nptel.ac.in/courses/105105157
- Bureau of Energy Efficiency (BEE), India: https://beeindia.gov.in
- IGBC (Indian Green Building Council): https://igbc.in
- openLCA (free LCA software): https://www.openlca.org
- EnergyPlus (free building energy simulation): https://energyplus.net

**Web Resources & NPTEL Links:**

- NPTEL: Sustainable Development – https://nptel.ac.in/courses/105105157
- NPTEL: Green Buildings – https://archive.nptel.ac.in/courses/105/102/105102195
- Bureau of Energy Efficiency (BEE), India: https://beeindia.gov.in
- IGBC (Indian Green Building Council): https://igbc.in
- openLCA (free LCA software): https://www.openlca.org
- EnergyPlus (free building energy simulation): https://energyplus.net
- OpenStudio (free building energy modelling): https://openstudio.net
- Global Carbon Project: https://www.globalcarbonproject.org

**Recommended Free & Open-Source AI/Computational Tools:**

- openLCA (free, open-source) – Life Cycle Assessment software for embodied carbon calculation
- EnergyPlus (free, US DOE) – Building energy simulation for operational energy analysis
- OpenStudio (free, NREL) – Building energy modeling interface for EnergyPlus
- Python (scikit-learn, Pandas, NumPy) on Google Colab – ML for energy prediction and material selection
- QGIS (free) – Green infrastructure planning and spatial sustainability analysis
- Ai-Tree Tools (free, USDA) – Urban tree carbon sequestration quantification
- SimaPro (student trial) – Life cycle assessment platform

**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:** Computational Thinking and Problem Solving with Python

**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:** Explain 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 | 1 |
| CO2 | 3 | 3 | 2 | 1 | 2 | | 1 | | | | 2 | 3 | 2 |
| CO3 | 3 | 3 | 2 | 1 | 2 | | 1 | | | | 1 | 3 | 2 |
| CO4 | 3 | 2 | 1 | 1 | 2 | | 1 | | | | 2 | 3 | 2 |
| CO5 | 3 | 2 | 1 | 1 | 2 | | 1 | | | | 2 | 3 | 2 |

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

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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 Resources:**

- NPTEL Computer Organization: https://nptel.ac.in
- NPTEL Digital Logic Design: https://nptel.ac.in
- Google Colab for Python: https://colab.research.google.com
- 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:** Basic knowledge of data structures, programming concepts, and problem-solving techniques. Familiarity with basic mathematics and discrete mathematical concepts.

**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 | 3 |
| 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:**

**Core 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.

Source: TB1: Chapters 1–4 (Cormen et al., CLRS 4th ed.); TB2: Chapter 5 (Kleinberg & Tardos)

**AI Integration:** Use VisuAlgo 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:**

**Core 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.

Source: TB1: Chapters 7–9, 28 (Cormen et al.); TB2: Chapter 5 (Kleinberg & Tardos)

**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:**

**Core 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.

Source: TB1: Chapters 16, 23–24 (Cormen et al.); TB2: Chapters 4, 6 (Kleinberg & Tardos)

**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:**

**Core 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.

Source: TB1: Chapters 15, 25 (Cormen et al.); TB2: Chapter 6 (Kleinberg & Tardos)

**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:**

**Core 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.
Source: TB1: Chapters 34–35 (Cormen et al.); TB2: Chapter 8 (Kleinberg & Tardos)
**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. TB1: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 4th Edition, MIT Press, 2022.
2. TB2: 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 Resources:**

- NPTEL – Design and Analysis of Algorithms, IIT Madras – nptel.ac.in
- MIT OpenCourseWare – Introduction to Algorithms (6.006) – ocw.mit.edu
- VisuAlgo – visualgo.net | Algorithm Visualizer – algorithm-visualizer.org
- 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 programming fundamentals and problem-solving techniques. Familiarity with basic concepts of data structures and object-oriented programming.

**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 | 3 | 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:**

**Core 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.

Source: TB1: Chapters 1–4 (Seth & Juneja, JAVA one step ahead); TB2: Chapters 1–3 (Samanta & Sarma)

**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:**

**Core 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.

Source: TB1: Chapters 5–7 (Seth & Juneja); TB3: Chapters 8–10 (Deitel & Deitel, Java 9 for Programmers)

**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:**

**Core 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.

Source: TB1: Chapters 8–11 (Seth & Juneja); TB2: Chapters 4–6 (Samanta & Sarma)

**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:**

**Core 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, TemporalAdjusters); 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).

Source: TB1: Chapters 12–14 (Seth & Juneja); TB3: Chapter 15 (Deitel & Deitel) — Java I/O and Files

**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:**

**Core Content:** String handling: CharSequence interface, String class, StringBuffer; 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. PreparedStatement, ResultSet; CRUD operations. JavaFX GUI: Scene Builder, App window structure, displaying text and images, event handling, node layout, mouse events.

Source: TB3: Chapters 11–12, 20 (Deitel & Deitel) — Strings, Multithreading, JDBC, JavaFX

**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. TB1: Anitha Seth and B.L. Juneja, JAVA one step ahead, Oxford University Press.
2. TB2: Debasis Samanta and Monalisa Sarma, Joy with JAVA — Fundamentals of Object-Oriented Programming, Cambridge University Press, 2023.
3. TB3: 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 Resources:**

- https://www.youtube.com/@JoyofLearningCSE/featured
- NPTEL – Programming in Java – nptel.ac.in/courses/106/105/106105191/
- Oracle Java Documentation – docs.oracle.com/en/java/
- 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 |
|---|---|---|---|---|---|---|---|
| 26PC04203P | EC | Digital Logic and Computer Organization Lab (Common to CSE, AI&DS, CSE(AI) & CSE(AI&ML)) | 0 | 0 | 0 | 2 | 1 |

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

**Course Objectives:**

- To understand the fundamental concepts of digital logic, Boolean algebra, and combinational and sequential circuits.
- To apply Boolean algebra, K-maps, and logic gates to design and simplify digital circuits.
- To analyze computer arithmetic operations, including signed arithmetic and IEEE 754 floating-point representation.
- To understand memory organization, cache mapping techniques, and I/O data transfer mechanisms.
- To apply simulation tools to analyze cache performance and compare programmed, interrupt-driven, and DMA-based I/O methods.

**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 | 3 | 2 |
| CO3 | 3 | 3 | 2 | 1 | 2 | | 3 | 3 | 3 | | 3 | 3 | 3 |
| CO4 | 2 | 3 | 1 | 2 | 2 | | 3 | 3 | 3 | | 2 | 3 | 3 |
| CO5 | 2 | 3 | 1 | 2 | 2 | | 3 | 3 | 3 | | 2 | 3 | 3 |

**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 Resources:**

- CircuitVerse – Online Digital Logic Simulator: https://circuitverse.org
- Logisim Evolution – Open Source Logic Simulator
- NPTEL Digital Logic Design: https://nptel.ac.in
- 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) & CSE(AI&ML)) | 0 | 0 | 0 | 3 | 1.5 |

**Pre-Requisites:** Data Structures, Programming for Problem Solving (Python/C/Java).

**Course Objectives:**

- To understand algorithm design paradigms including divide and conquer, greedy, dynamic programming, and approximation methods.
- To develop the ability to analyse algorithm correctness and efficiency using asymptotic notation, recurrence relations, and amortized analysis.
- To design efficient algorithmic solutions for optimization and computational problems.
- To apply advanced algorithmic strategies including graph algorithms, backtracking, and branch-and-bound.
- To utilize AI-assisted tools for visualization, implementation, complexity analysis, and performance evaluation of algorithms.

**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 for sorting, searching, and matrix multiplication. (L4)

**CO3:** Apply greedy algorithms to graph and optimization problems. (L3)

**CO4:** Design dynamic programming solutions for classical optimization problems. (L4)

**CO5:** Apply complexity classification, backtracking, branch-and-bound, and approximation methods to computationally difficult 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 | 1 | 3 | 2 | | 3 | 3 | 3 | | 1 | 3 | 3 |
| 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**

**Module I: Algorithm Analysis and Mathematical Foundations**

**Topics**: Asymptotic growth rates, recurrence relations, recursion tracing, and self-balancing tree structures, building the mathematical foundation needed for algorithm analysis.

**AI Integration**: AI tools are used to visualize complexity growth trends, verify asymptotic notation, derive and validate recurrence solutions against the Master theorem, generate recursion trees, and visualize AVL/B-Tree balancing operations.

**Suggested Tools:** VisuAlgo; Python Tutor; Wolfram Alpha; SymPy; ChatGPT

**Experiments**:

- **Experiment 1: Complexity Growth Comparison** Implement algorithms representing O(log n), O(n), O(n log n), and $O(n^2)$ growth, execute them for varying input sizes, plot execution time, and compare practical versus theoretical growth using VisuAlgo and Python Tutor.
- **Experiment 2: Recurrence Relation Verification** Solve recurrence equations such as $T(n)=T(n-1)+1$, $T(n)=2T(n/2)+n$, and $T(n)=T(n/2)+1$, and verify the solutions experimentally, using Wolfram Alpha, SymPy, and ChatGPT to generate derivations and validate Master theorem solutions.
- **Experiment 3: Recursion Tree Visualization** Implement Merge Sort and Binary Search, trace their recursive execution, compare runtime, and generate recursion trees using VisuAlgo and Python Tutor.
- **Experiment 4: AVL Tree and B Tree Analysis** Implement AVL tree insertion with rotations and B Tree insertion, compare balancing operations, and use VisuAlgo and ChatGPT to generate insertion sequences and visualize balancing.

**Module II: Divide and Conquer**

**Topics:** Divide-and-conquer strategy applied to sorting, order-statistics selection, and matrix multiplication, with emphasis on comparing recursive decomposition and measured performance.

**AI Integration**: AI tools help compare pivot strategies, generate test datasets and edge cases, compare expected versus observed complexity, and trace the recursive decomposition used in matrix multiplication.

**Suggested Tools:** VisuAlgo; Algorithm Visualizer; ChatGPT

**Experiments:**

- **Experiment 5: Sorting Algorithms Comparison** Implement Merge Sort, Quick Sort, and Heap Sort, and compare their runtime, number of comparisons, and swaps, using AI tools to compare pivot strategies and generate datasets.
- **Experiment 6: Selection Algorithms** Implement Randomized Selection and Deterministic Selection to find the median and kth smallest element, using AI tools to generate edge cases and compare expected complexity.
- **Experiment 7: Matrix Multiplication Optimization** Implement Classical Matrix Multiplication and Strassen's Algorithm, compare execution time, and use AI tools to trace recursive decomposition and compare multiplication counts.

**Module III: Greedy Algorithms**

**Topics:** Greedy algorithmic strategy applied to interval scheduling, minimum spanning trees, shortest paths, and optimal prefix coding.

**AI Integration:** AI tools generate scheduling scenarios and character-frequency datasets, visualize graph traversal, and compare optimal schedules, path selection, and compression ratios.

**Suggested Tools:** ChatGPT; VisuAlgo; Gephi; Cytoscape

**Experiments:**

- **Experiment 8: Activity Selection and Scheduling** Implement Activity Selection, Fractional Knapsack, and Job Scheduling, using AI tools to generate scheduling scenarios and compare optimal schedules.
- **Experiment 9: Minimum Spanning Tree and Shortest Path** Implement Prim's Algorithm, Kruskal's Algorithm, and Dijkstra's Algorithm, using Gephi and Cytoscape alongside VisuAlgo to visualize graph traversal and compare path selection.

- **Experiment 10: Huffman Encoding** Generate character frequencies, construct a Huffman Tree, and calculate the compression ratio, using AI tools to generate frequency datasets and compare compression.

**Module IV: Dynamic Programming**

**Topics:** Dynamic programming formulation of classical optimization problems, including DP table construction, all-pairs shortest paths, and comparison of recursive, memoized, and tabulated implementations.

**AI Integration:** AI tools generate DP tables, trace solution reconstruction, visualize optimized paths, and compare route optimization along with runtime and memory trade-offs across implementation strategies.

**Suggested Tools:** DP Visualizer; Python Tutor; ChatGPT

**Experiments:**

- **Experiment 11: Dynamic Programming Table Construction** Implement Longest Common Subsequence, 0/1 Knapsack, and Edit Distance, using AI tools to generate DP tables and trace solution reconstruction.
- **Experiment 12: Shortest Path and Travelling Salesman** Implement the Floyd–Warshall algorithm and a Travelling Salesman solution, using AI tools to visualize paths and compare route optimization.
- **Experiment 13: Memorization vs Tabulation** Compare recursive, memorized, and tabulated implementations by measuring runtime and memory, using AI tools to generate execution comparisons.

**Module V: Advanced Topics and Computational Complexity**

**Topics:** Computationally hard problems addressed through backtracking and branch-and-bound, complexity classification (P, NP, NP-Complete, NP-Hard), and approximation algorithms.

**AI Integration:** AI tools visualize state-space trees, compare pruning strategies, generate reduction examples, and compare exact versus approximate solution quality.

**Suggested Tools:** Graphviz; Gephi; Claude; ChatGPT

**Experiments:**

- **Experiment 14: Backtracking and Branch-and-Bound** Implement the N Queens problem, Graph Coloring, and TSP using backtracking and branch-and-bound, using AI tools to visualize state-space trees and compare pruning.
- **Experiment 15: Complexity Classification and Approximation** Classify problems into P, NP, NP-Complete, and NP-Hard, and implement Vertex Cover Approximation and TSP Approximation, using AI tools to generate reduction examples and compare exact and approximate solutions.

**Text Books:**

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

**Reference Books:**

1. Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani, Algorithms.
2. Robert Sedgewick and Kevin Wayne, Algorithms.

3. Steven S. Skiena, The Algorithm Design Manual.
4. Sara Baase and Allen Van Gelder, Computer Algorithms.

**Online Learning Resources:**

- NPTEL – Design and Analysis of Algorithms – nptel.ac.in
- MIT OpenCourseWare – Introduction to Algorithms – ocw.mit.edu
- VisuAlgo – visualgo.net
- Algorithm Visualizer – algorithm-visualizer.org
- Gephi – gephi.org
- Python Tutor – pythontutor.com
- Wolfram Alpha – wolframalpha.com

**Recommended Free & Open-Source AI / Computational Tools**

- Visualization Tools: VisuAlgo; Algorithm Visualizer; DP Visualizer; Graphviz; Gephi; Cytoscape – used to visualize algorithm execution, recursion trees, DP tables, and graph / state-space structures.
- AI Assistants: ChatGPT; Claude – used to generate derivations, test cases, reduction examples, and comparative complexity analysis.
- Mathematical & Tracing Tools: Wolfram Alpha; SymPy; Python Tutor – used to solve and verify recurrence relations and trace program execution step-by-step.

**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:** Programming for Problem Solving and Data Structures

**Course Objectives:**

- To develop practical skills in object-oriented analysis, design, and implementation using Java programming constructs and software design principles.
- To apply classes, objects, inheritance, interfaces, exception handling, collections, and multithreading to build modular and reusable applications.
- To design and implement Java applications using packages, file handling, Java I/O, JDBC connectivity, and GUI development.
- To utilize AI-assisted development tools for program generation, debugging, UML modelling, testing, and code optimization.
- To build complete object-oriented applications following software engineering and intelligent development practices.

**Course Outcomes (COs):**

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

**CO1:** Develop Java programs using object-oriented principles, data types, operators, and control structures. (L3)

**CO2:** Design and implement classes, methods, inheritance, interfaces, and collection framework-based applications. (L4)

**CO3:** Apply exception handling, packages, file operations, multithreading, and JDBC connectivity to develop robust Java applications. (L3)

**CO4:** Build GUI-based applications and interactive software solutions using JavaFX and modern Java APIs. (L4)

**CO5:** Utilize AI-assisted software development tools for UML modelling, code generation, testing, debugging, and application enhancement (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 | 3 | 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 | 2 | 2 | 2 | 2 | | 3 | 3 | 3 | | 1 | 2 | 3 |
| CO5 | 3 | 2 | 2 | 1 | 2 | | 3 | 3 | 3 | | 1 | 2 | 3 |

**List of Experiments**

**Module I: OOP Fundamentals, Data Types, Operators & Control Structures**

**Topics:** Primitive data types, operators, precedence, associativity, and type conversion; conditional and nested statements, switch, loops, and control-transfer statements; searching, sorting, and matrix-based applications using arrays.

**AI Integration:** Use ChatGPT or GitHub Copilot for program generation, test-case generation, execution tracing, and algorithm validation; use draw.io to generate flowcharts.

**Suggested Tools:** Java (JDK), VS Code / IntelliJ IDEA, ChatGPT, GitHub Copilot, draw.io

**Experiments:**

- **Experiment 1: Data Types & Operators** — Develop Java programs to demonstrate primitive data types, operators, precedence, associativity, and type conversion.
- **Experiment 2: Control Structures** — Develop Java applications using conditional statements, nested conditions, switch statements, loops, and control-transfer statements.
- **Experiment 3: Arrays, Searching & Sorting** — Implement searching, sorting, and matrix-based applications using arrays and control structures.

**Module II: Classes, Objects, Methods & Collections**

**Topics:** Classes, objects, constructors, constructor and method overloading, recursion; access control, nested classes, static and final members; Java Collection Framework — ArrayList, LinkedList, HashMap, HashSet, TreeMap.

**AI Integration:** Use StarUML and draw.io to generate UML class diagrams; use AI tools to generate class structures and recommend suitable collections.

**Suggested Tools:** Java (JDK), StarUML, draw.io, ChatGPT

**Experiments:**

- **Experiment 4: Classes & Constructors** — Develop Java applications using classes, objects, constructors, constructor overloading, method overloading, and recursive methods.
- **Experiment 5: Access Control & Members** — Design real-world object-oriented applications using access control, nested classes, static members, and final members.
- **Experiment 6: Collection Framework** — Implement and compare applications using the Java Collection Framework including ArrayList, LinkedList, HashMap, HashSet, and TreeMap.

**Module III: Arrays, Inheritance & Interfaces**

**Topics:** One- and multi-dimensional arrays, sorting, searching, and array-manipulation techniques; inheritance — single, multilevel, hierarchical, method overriding, dynamic dispatch; interfaces, interface inheritance, abstract classes, and functional interfaces.

**AI Integration:** Use AI tools to generate inheritance hierarchies and compare interface-based and abstract-class-based designs.

**Suggested Tools:** Java (JDK), StarUML, ChatGPT, GitHub Copilot

**Experiments:**

- **Experiment 7: Array Techniques** — Develop Java programs using one-dimensional and multi-dimensional arrays, sorting, searching, and array-manipulation techniques.
- **Experiment 8: Inheritance** — Implement inheritance concepts including single, multilevel, and hierarchical inheritance, method overriding, and dynamic dispatch.
- **Experiment 9: Interfaces & Abstract Classes** — Design applications using interfaces, interface inheritance, abstract classes, and functional interfaces.

**Module IV: Packages, Exception Handling & Java I/O**

**Topics:** User-defined packages and multi-package applications; exception handling — try-catch-finally, throw, throws, multiple catch, user-defined exceptions, try-with-resources; file handling using Scanner, byte streams, character streams, and the Java Files API.

**AI Integration:** Use AI tools for debugging, exception analysis, package-architecture generation, and file-operation optimization.

**Suggested Tools:** Java (JDK), VS Code, ChatGPT, GitHub Copilot

**Experiments:**

- **Experiment 10: Packages** — Create and implement user-defined packages and develop multi-package applications.
- **Experiment 11: Exception Handling** — Develop Java programs using exception-handling mechanisms including try-catch-finally, throw, throws, multiple catch blocks, user-defined exceptions, and try-with-resources.
- **Experiment 12: File Handling** — Implement Java file-handling applications using Scanner, byte streams, character streams, and the Java Files API.

**Module V: Strings, Multithreading, JDBC & JavaFX**

**Topics:** String, StringBuffer, StringBuilder, and string-manipulation techniques; multithreading — Thread class, Runnable interface, synchronization, producer-consumer model, deadlock simulation; JDBC connectivity with CRUD, PreparedStatement, ResultSet; JavaFX GUI design and event handling.

**AI Integration**: Use AI tools for multithread debugging, GUI generation, SQL generation, and intelligent code optimization.

**Suggested Tools:** Java (JDK), JavaFX, MySQL, ChatGPT

**Experiments:**

- **Experiment 13: String Handling** — Develop applications using String, StringBuffer, StringBuilder, and string-manipulation techniques.
- **Experiment 14: Multithreading** — Implement multithreaded Java applications using the Thread class, Runnable interface, synchronization, the producer-consumer model, and deadlock simulation.
- **Experiment 15: JDBC & JavaFX Integration** — Develop integrated Java applications using JDBC database connectivity, CRUD operations, PreparedStatement and ResultSet, and JavaFX GUI design with event handling.

**Text Books:**

1. Anitha Seth & B. L. Juneja, JAVA One Step Ahead, Oxford University Press.
2. Debasis Samanta & Monalisa Sarma, Joy with JAVA – Fundamentals of Object-Oriented Programming, Cambridge University Press, 2023.
3. Paul Deitel & Harvey Deitel, Java 9 for Programmers, 4th ed., Pearson Education.

**Reference Books:**

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

**Online Learning Resources:**

- NPTEL – Programming in Java – nptel.ac.in
- Oracle Java Documentation – docs.oracle.com/en/java
- StarUML Documentation – staruml.io
- draw.io Documentation – draw.io
- GitHub Copilot Documentation – github.com/features/co-pilot
- https://www.youtube.com/@JoyofLearningCSE/featured

**Recommended Free & Open-Source Java / AI Tools**

- Language & IDE: Java (JDK), VS Code, IntelliJ IDEA – development and debugging.
- UML & Diagrams: StarUML, draw.io – class, inheritance and flow diagrams.

- Database & GUI: MySQL, JavaFX (Scene Builder) – JDBC connectivity and interfaces.
- AI Assistants: ChatGPT, GitHub Copilot – code generation, debugging, test generation and optimization.

**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(AI&ML)) | 1 | 0 | 0 | 2 | 2 |

**Pre-Requisites:** Basic programming knowledge in any Programming language.

**Course Objectives**

- To introduce students to modern web application development using frontend, backend, and database technologies.
- To enable students to utilize 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 work flows. (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 | 3 | 3 |
| CO4 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | | 2 | 3 | 3 |
| CO5 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | | 2 | 2 | 3 |

**List of Experiments**

**Module I: Frontend Development with HTML & CSS (Weeks 1–4)**

**Topics:** HTML5 semantic elements — text, images, hyperlinks, lists, tables, multimedia; responsive web forms and validation interfaces; CSS selectors, typography, colors, Flexbox, Grid, and the CSS Box Model; reusable UI components — navigation bars, cards, dashboards, responsive menus.

**AI Integration:** HTML generation and UI refinement; form design and responsive styling; CSS generation and layout optimization; component generation and UI enhancement.

**Suggested Tools:** HTML5, CSS3, VS Code

**Experiments:**

- **Week 1: AI-Assisted Web Page Design using HTML** — Develop a personal portfolio website using HTML5 semantic elements including text, images, hyperlinks, lists, tables, and multimedia; use AI tools to generate and refine webpage structure and improve layout.
- **Week 2: Responsive Web Forms using HTML and CSS** — Design responsive registration and login forms with appropriate input controls and a validation interface; generate and optimize form layout using AI assistance.
- **Week 3: CSS Styling and Responsive Layout Development** — Develop responsive webpages using CSS selectors, typography, colors, Flexbox, Grid, and the CSS Box Model; compare manually designed and AI-generated layouts.
- **Week 4: User Interface Component Development** — Create reusable UI components including navigation bars, cards, dashboards, and responsive menus; use AI tools to improve UI consistency and responsiveness.

**Module II: Client-Side Programming with JavaScript (Weeks 5–8)**

**Topics:** JavaScript ES6 fundamentals — variables, operators, loops, functions, arrays, objects; DOM manipulation and event handling; client-side form validation; browser storage (Local Storage, Session Storage); API integration and dynamic content using the Fetch API and JSON.

**AI Integration:** Code generation and debugging; event-logic generation; validation generation and code correction; API integration support and response handling.

**Suggested Tools:** JavaScript ES6, Browser Console, REST APIs

**Experiments:**

- **Week 5: JavaScript Programming Fundamentals** — Develop interactive web pages using variables, operators, loops, functions, arrays, and objects; use AI assistance to generate and optimize scripts.
- **Week 6: DOM Manipulation and Event Handling** — Develop event-driven applications using DOM operations, event listeners, dynamic content updates, and user-interaction handling.
- **Week 7: Client-side Form Validation and Browser Storage** — Implement form validation using JavaScript and store user information using Local Storage and Session Storage.
- **Week 8: API Integration and Dynamic Content Generation** — Consume external APIs and dynamically update webpage content using the Fetch API and JSON processing.

**Module III: Backend Development with Node.js & Express (Weeks 9–11)**

**Topics:** Backend server setup with routing and middleware; RESTful API development with CRUD operations; database integration with MongoDB; user authentication and authorization.

**AI Integration:** Server generation and debugging; API generation and testing; query generation and schema optimization.

**Suggested Tools:** Node.js, Express.js, MongoDB, Postman

**Experiments**:

- **Week 9: Backend Application Setup using Node.js and Express.js** — Create a backend server application with routing and middleware support; generate boilerplate backend code using AI tools.
- **Week 10: REST API Development and CRUD Operations** — Develop RESTful APIs supporting Create, Read, Update, and Delete operations.

- **Week 11: Database Integration and Authentication** — Connect backend applications with MongoDB and implement user authentication and authorization mechanisms.

**Module IV: Full-Stack Integration & Deployment (Weeks 12–14)**
**Topics:** Frontend–backend integration; complete application workflows; full-stack testing and performance optimization; documentation; cloud deployment and application monitoring.
**AI Integration:** Integration support and issue resolution; test generation and optimization; deployment assistance and troubleshooting.
**Suggested Tools:** React.js, Node.js, REST APIs, GitHub
**Experiments:**

- **Week 12: Frontend–Backend Integration** — Integrate frontend interfaces with backend APIs and implement complete application workflows.
- **Week 13: Full-Stack Application Testing and Optimization** — Test application functionality, improve performance, and generate documentation using AI assistance.
- **Week 14: Deployment and Application Monitoring** — Deploy applications to cloud platforms and monitor application behavior and performance.

**Module V: AI-Assisted Web Application Mini Project (Week 15)**
**Topics:** End-to-end MERN-stack application development integrating frontend, backend, database, testing, deployment, and documentation.
**AI Integration:** Code generation, debugging, testing, documentation, and deployment support.
**Suggested Tools:** MERN Stack (MongoDB, Express.js, React.js, Node.js), GitHub
**Experiments:**

- **Week 15: AI-Assisted Web Application Mini Project** — Develop and demonstrate a complete web application integrating frontend, backend, database, testing, deployment, and documentation.

**Suggested Mini Projects:**

1. Student Management System
2. Smart Attendance System
3. Event Registration Portal
4. Expense Tracker
5. Inventory Management System
6. Task Management Dashboard
7. Online Quiz Application
8. Campus Information Portal

**Text Books**:

1. Jon Duckett, HTML and CSS: Design and Build Websites, Wiley Publications.
2. David Flanagan, JavaScript: The Definitive Guide, O'Reilly Media.
3. Vasan Subramanian, Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React and Node, Apress.

**Reference Books:**

1. Robert W. Sebesta, Programming the World Wide Web, Pearson Education.
2. Ethan Brown, Web Development with Node and Express, O'Reilly Media.
3. Andrew Ng, Generative AI for Everyone, DeepLearning.AI.

**Online Learning Resources:**

- MDN Web Docs – HTML, CSS & JavaScript Documentation – developer.mozilla.org
- W3Schools – Web Development Tutorials – w3schools.com
- React Documentation – react.dev
- Node.js Documentation – nodejs.org/docs
- Express.js Documentation – expressjs.com
- MongoDB Documentation – mongodb.com/docs
- GitHub Skills & GitHub Education – skills.github.com
- Visual Studio Code Documentation – code.visualstudio.com/docs
- OpenAI Developer Resources – platform.openai.com/docs
- freeCodeCamp Web Development – freecodecamp.org
- https://www.youtube.com/@JoyofLearningCSE/featured

**Recommended Free & Open-Source AI / Web Development Tools:**

- Frontend: HTML5, CSS3, React.js – responsive interfaces and reusable components.
- Backend & Database: Node.js, Express.js, MongoDB – server-side logic and data storage.
- Tooling: VS Code, Postman, GitHub – editing, API testing and version control.
- AI Assistants: ChatGPT, GitHub Copilot – code generation, debugging, testing and documentation.

## B.Tech. – II Year I 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:** No specific prerequisites; open to all engineering branches

**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, define key concepts of scope and public awareness, and describe the uses, over-exploitation and associated environmental problems of forest, water, mineral, food and energy resources with reference to Indian and global case studies. (L4)

**CO2:** Describe the structure and function of different ecosystem types, explain ecological succession, food chains and ecological pyramids, and outline the concept and classification of biodiversity, its value, threats and conservation strategies (in-situ and ex-situ) including India's status as a mega-diversity nation. (L3)

**CO3:** Evaluate the causes, effects and control measures of air, water, soil, marine, noise, thermal and nuclear pollution; apply principles of solid waste management for urban and industrial wastes; analyze pollution case studies and evaluate disaster management strategies for floods, earthquakes, cyclones and landslides. (L5)

**CO4:** Analyze social issues related to unsustainable development; evaluate the impacts of climate change, global warming, acid rain and ozone layer depletion; examine Indian environmental legislation (EPA, Air Act, Water Act, Wildlife Protection Act, Forest Conservation Act) and assess their enforcement challenges. (L5)

**CO5:** Describe population growth trends and their environmental impacts; identify links between environment and human health; demonstrate the role of information technology and AI-based tools in environmental monitoring, public health surveillance and field data collection; apply digital tools (QGIS, Python, iNaturalist) for environmental documentation. (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 | 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 | 2 | 2 |
| CO4 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | | | | 3 | 2 | 2 |
| CO5 | 1 | 1 | | 1 | 3 | 3 | 2 | | | | 3 | 2 | 2 |

**Unit I: Multidisciplinary Nature of Environmental Studies and Natural Resources:**

**Core Content:** 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 srces, 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:**

**Core Content:** 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:**

**Core Conten**t: 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:**

**Core Content:** 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).

**Free / Open-Source AI Tools:** Python (xarray, Matplotlib, scikit-learn) – Google Colab (free); CMIP6 climate data (free, ESGF portal); QGIS (free); openLCA (free, LCA software).

**Unit V: Human Population and the Environment:**

**Core Content:** 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).

**Free / Open-Source AI Tools:** iNaturalist (free); eBird (free); QGIS (free); Python (scikit-learn, TensorFlow) – Google Colab (free); OpenStreetMap (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.

**Web Resources & Online Platforms:**

- https://www.coursera.org/learn/python-for-applied-data-science-ai
- https://www.coursera.org/learn/python?specialization=python#syllabus
- NPTEL: Environmental Science – https://nptel.ac.in/courses/124107002
- Google Earth Engine (free cloud geospatial platform) – https://earthengine.google.com
- Global Forest Watch (free deforestation monitoring) – https://www.globalforestwatch.org
- OpenAQ (free global air quality open data) – https://openaq.org
- iNaturalist (free citizen science biodiversity app) – https://www.inaturalist.org
- QGIS (free, open-source GIS) – https://qgis.org
- India Meteorological Department (free data) – https://mausam.imd.gov.in
- Central Pollution Control Board (CPCB) data portal (free) – https://cpcb.nic.in

**Recommended Free & Open-Source AI / Digital Tools:**

- Google Earth Engine (free cloud platform) – real-time satellite data analysis for deforestation, land use and vegetation change detection
- QGIS (free, open-source GIS) – environmental mapping, watershed delineation, disaster risk zone mapping, biodiversity hotspot analysis
- Python (NumPy, Pandas, Matplotlib, scikit-learn, TensorFlow) on Google Colab (free) – AQI prediction, population growth ML models, climate data analysis
- iNaturalist (free, open-source citizen science) – species identification using image recognition during field work; community biodiversity database
- eBird (free, Cornell Lab) – bird species recording and mapping during field visits; global bird distribution data
- OpenAQ (free, open-source) – global real-time and historical air quality data for ML model training
- MaxEnt (free) – species distribution modelling for biodiversity hotspot prediction from occurrence data
- Global Forest Watch (free dashboard) – monitoring deforestation and forest cover change using satellite imagery
- CMIP6 climate datasets (free, ESGF portal) – global climate model data for Python-based climate change scenario analysis

# B. Tech. – II Year
# II Semester

## B.Tech. – II Year II 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 knowledge of biology and general science concepts. Familiarity with fundamental computer applications and problem-solving techniques.

**Course Objectives:**

- To provide foundational biological knowledge with engineering relevance.
- To enable teachers to demonstrate AI-driven applications in biology and bioengineering.
- To encourage interdisciplinary teaching that connects biology, engineering, and computational tools.
- To prepare students for innovation in healthcare, materials, and bio-industries.

**Course Outcomes (COs):**

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

**CO1:** Apply the fundamentals of biology, biomolecules, and human systems relevant to engineering applications. (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. (L4)

**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 | 2 |
| CO3 | 2 | 3 | 2 | 3 | 3 | 2 | 1 | | | | 3 | 2 | 1 |
| CO4 | 2 | 3 | 2 | 3 | 3 | 2 | 1 | | | | 3 | 2 | 2 |
| CO5 | 2 | 3 | 3 | 3 | 3 | 2 | 1 | | | | 3 | 2 | 2 |

**Unit I: Foundations of Biology and AI:**

**Core 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 AlphaFold, Use of DeepChem to simulate enzyme-substrate binding, Compare AI-predicted protein structures with experimental data.

**Unit II: Biomolecules in Engineering Applications:**

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

**AI Integration:** AI-driven generative design for biomaterials, Lipids in biodiesel &detergents.

**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:**
**Core 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:**
**Core 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:**
**Core 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., Thilagaraj W.,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.

**Web links and Video Lectures (e-Resources):**

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

**AI Tools for the Course:**

- DeepChem – drug discovery simulations.
- AlphaFold – protein structure prediction.
- Generative Design AI – biomaterial innovation.
- ANSYS – biomimetic engineering simulations.
- MATLAB Simulink – biological system modeling.
- AI Medical Imaging Tools – diagnostics and healthcare.

**B.Tech. – II Year II 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 human values, ethics, and social responsibilities. Familiarity with fundamental concepts of personal, family, and professional relationships.

**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 concept of universal human values, natural acceptance, and the purpose of human life as articulated in Indian value philosophy. (L3)

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

**CO3:** Analyse 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:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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:**

**Core 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 Resources:**

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

**YouTube / Video Resources:**

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

## B.Tech. – II Year II Semester

| Course Code | Category | Name of the Course | L | T | Pr | P | C |
|---|---|---|---|---|---|---|---|
| 26PC05204T | PC | Operating Systems and Intelligent Resource Management | 3 | 0 | 0 | 0 | 3 |

**Pre-Requisites:** **B**asic knowledge of computer organization and programming fundamentals. Familiarity with data structures, algorithms, and basic computer architecture concepts.

**Course Objectives:**

- Understand the structure and functions of operating systems.
- Analyze process management, scheduling, and concurrency mechanisms.
- Apply memory management, file systems, and deadlock handling techniques.
- Explore AI-based approaches for resource management in operating systems.
- Understand modern operating system trends including cloud and container systems.

**Course Outcomes (COs):**

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

**CO1:** Apply OS structure, services, and system calls. (L3)

**CO2:** Analyze process scheduling and concurrency mechanisms. (L4)

**CO3:** Apply memory management, file systems, and deadlock handling techniques. (L3)

**CO4:** Evaluate AI-based approaches for intelligent resource management. (L5)

**CO5:** Analyze modern OS architectures including virtualization and cloud systems. (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 | | 1 | | | | 2 | 3 | 2 |
| CO2 | 3 | 3 | 2 | 2 | 2 | | 1 | | | | 2 | 3 | 3 |
| CO3 | 3 | 3 | 2 | 2 | 2 | | 1 | | | | 2 | 3 | 3 |
| CO4 | 3 | 3 | 3 | 3 | 3 | | 1 | | | | 3 | 3 | 3 |
| CO5 | 3 | 3 | 2 | 2 | 3 | | 1 | | | | 3 | 3 | 3 |

**Unit I: Operating Systems Fundamentals, System Structures & Process Management:**

**Core Content:** Operating Systems Overview: definition and role of an operating system; functions of OS — process management, memory management, file systems, I/O systems; operating system operations and services; types of operating systems — batch, time-sharing, real-time, distributed, mobile; computing environments — traditional, mobile, distributed, cloud; open-source operating systems and their significance. System Structures: OS services and system programs; user interface — CLI and GUI; system calls — concept and types (process control, file management, device management, information maintenance, communication); system programs and utilities; OS design and implementation approaches; OS structures — monolithic, layered, modular; system boot process; OS debugging techniques. Process Management: process concept and lifecycle; Process Control Block (PCB); process scheduling concepts; scheduling criteria (CPU utilization, throughput, turnaround time, waiting time, response time); scheduling algorithms — FCFS, SJF, Priority, Round Robin; multiprocessor scheduling; client–server communication models.

**AI Integration:** Use ChatGPT or Claude to simulate process scheduling step-by-step and system-call workflows; use GitHub Copilot to implement scheduling algorithms in Python; use AI-based diagram tools to visualize process lifecycle and OS architecture.

**Suggested Tools:** ChatGPT, Claude, GitHub Copilot, Python

**Unit II: Concurrency, Multithreading & Memory Management:**

**Core Content:** Concurrency & Multithreading: process vs. thread; multithreading models — many-to-one, one-to-one, many-to-many; thread libraries and threading issues; inter-process communication (IPC) — shared memory, message passing. Synchronization: race conditions and the critical-section problem; synchronization solutions — mutex locks, semaphores, monitors; busy waiting and blocking; classical synchronization problems — Dining Philosophers, Readers–Writers. Memory Management: memory hierarchy and allocation; swapping; contiguous memory allocation; paging and segmentation. Virtual Memory: concept of virtual memory; demand paging; copy-on-write; page replacement algorithms — FIFO, LRU, Optimal; frame allocation strategies; thrashing; memory-mapped files; kernel memory allocation.

**AI Integration:** Use Google Colab and Python to simulate page replacement algorithms; use ChatGPT to debug synchronization problems and generate solutions for concurrency issues; use Python Tutor and visualizers to visualize thread execution and memory allocation.

**Suggested Tools:** Google Colab, Python, ChatGPT, Python Tutor

**Unit III: Deadlocks, File Systems, Storage & Security:**

**Core Content:** Deadlocks: system model and resource allocation; deadlock conditions — mutual exclusion, hold and wait, no preemption, circular wait; deadlock prevention; deadlock avoidance (Banker's Algorithm); deadlock detection and recovery. File Systems: file concepts and attributes; file operations; directory structures — single-level, two-level, tree; file system implementation — allocation methods, directory implementation. Secondary Storage: disk structure and disk attachment; disk scheduling algorithms — FCFS, SSTF, SCAN, C-SCAN. Protection & Security: goals and principles of protection; access matrix and access control mechanisms; revocation of access rights; system security — program threats (virus, worms, trojans), system and network threats, cryptography basics, user authentication methods, firewalls and intrusion detection.

**AI Integration:** Use Linux tools for file-system and permission handling; use Wireshark and log tools to analyze system/network behavior; use ChatGPT to simulate deadlock scenarios and generate security case studies.

**Suggested Tools:** Linux, Wireshark, ChatGPT

**Unit IV: Intelligent Operating Systems & AI-Based Resource Management:**

**Core Content:** Introduction to AI in operating systems; intelligent CPU scheduling using machine learning; predictive resource allocation techniques; reinforcement learning for scheduling decisions; adaptive memory management; workload prediction and performance optimization; self-healing and self-optimizing operating systems; AI-driven cloud resource orchestration.

**AI Integration:** Use Scikit-learn to build workload prediction models; use Google Colab to train and test ML models; use ChatGPT to generate datasets and explain model behavior.

**Suggested Tools:** Scikit-learn, Google Colab, ChatGPT

**Unit V: Modern Operating Systems, Virtualization & Cloud Systems:**

**Core Content:** Evolution of modern operating systems; virtualization concepts and hypervisors; containerization — Docker architecture and usage; container orchestration — Kubernetes basics; cloud operating systems and service models; edge and distributed operating systems; microkernel vs. monolithic vs. hybrid architectures; security in modern OS environments.

**AI Integration:** Use Docker for container creation and deployment; use Kubernetes for basic orchestration and scheduling; use ChatGPT to generate Dockerfiles and troubleshoot deployment.

**Suggested Tools:** Docker, Kubernetes, ChatGPT

**Text Books:**

1. A. Silberschatz, P. B. Galvin & G. Gagne, Operating System Concepts, 10th ed., Wiley, 2018.
2. A. S. Tanenbaum & H. Bos, Modern Operating Systems, 4th ed., Pearson, 2015.
3. W. Stallings, Operating Systems: Internals and Design Principles, 9th ed., Pearson, 2018.

**Reference Books:**

1. D. M. Dhamdhere, Operating Systems: A Concept-Based Approach, 3rd ed., McGraw-Hill, 2017.
2. A. S. Tanenbaum & A. S. Woodhull, Operating Systems: Design and Implementation, 3rd ed., Pearson, 2006.
3. A. S. Tanenbaum & M. Van Steen, Distributed Systems: Principles and Paradigms, 2nd ed., Pearson, 2017.
4. T. Erl, R. Puttini & Z. Mahmood, Cloud Computing: Concepts, Technology & Architecture, Prentice Hall, 2013.

**Web Resources:**

- NPTEL – Operating Systems – nptel.ac.in/courses/106106144
- Docker Documentation – docs.docker.com
- Kubernetes Documentation – kubernetes.io
- Linux Foundation Training – training.linuxfoundation.org
- Coursera – Operating Systems and Systems Programming – coursera.org
- https://www.youtube.com/@JoyofLearningCSE/featured

**Recommended Free & Open-Source OS / AI Tools:**

- Operating System: Linux (Ubuntu) – process, memory and file-system experimentation.
- Programming & ML: Python, Scikit-learn – algorithm implementation and workload prediction.
- Containers & Cloud: Docker, Kubernetes – virtualization and orchestration.
- Analysis: Wireshark, Python Tutor – system/network and execution visualization.
- AI Assistants: ChatGPT, Claude, GitHub Copilot – simulation, debugging and Dockerfile generation.
- Notebooks: Google Colab – ML model training and testing.

## 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:** Data Structures and basic programming in Python.

**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.. (L3)

**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 | 3 | 3 |
| CO2 | 3 | 3 | 2 | 2 | 2 | | 1 | | | | 1 | 3 | 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:**

**Core 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; centralized vs. distributed and cloud database systems; overview of NoSQL categories. (Source: TB1 Ch.1–3, 7–9; TB2 Ch.1–3, 7–9)

**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.

**Suggested Tools:** draw.io; MySQL Workbench; ChatGPT

**Unit II: Relational Model and SQL:**

**Core Content:** Relational model: relations, tuples, attributes, domains, schemas; relational algebra – selection, projection, 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. (Source: TB1 Ch.2–6, 11; TB2 Ch.5–6, 14–15)

**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.

**Suggested Tools:** MySQL; SQLite; ChatGPT; GitHub Copilot; Google Colab

**Unit III: Database Design and Normalization:**

**Core 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. (Source: TB1 Ch.7–8; TB2 Ch.14–15)

**AI Integration:** Use ChatGPT to input a schema and obtain normalized tables; validate normalization steps manually; use Jupyter Notebook for FD closure calculations; practice AI-assisted schema design and verify with Armstrong's axioms.

**Suggested Tools:** ChatGPT; Jupyter Notebook

**Unit IV: Transaction Management and Concurrency Control:**

**Core 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, 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). (Source: TB1 Ch.14–17; TB2 Ch.19–21)

**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 them.

**Suggested Tools:** Python; ChatGPT

**Unit V: Intelligent Data Management for AI Systems:**

**Core Content:** NoSQL systems: motivation (3Vs, schema flexibility, horizontal scale); CAP theorem; document stores (MongoDB), key-value (Redis), column-family (Cassandra), graph (Neo4j – Cypher). Text-to-SQL and LLM-over-database: natural-language-to-SQL translation; schema linking; accuracy challenges; RAG-over-SQL; 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; re-ranking. Data governance: data catalog (Apache Atlas), lineage tracking; GDPR Article 5; India DPDP Act 2023. (Source: TB1 Ch.10, 22–23; TB2 Ch.24–25; Kleppmann)

**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.

**Suggested Tools:** ChatGPT; Claude; MongoDB Atlas; FAISS; Pinecone

**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 Resources:**

- NPTEL – Database Management Systems, IIT Madras – nptel.ac.in
- MySQL – dev.mysql.com/doc | MongoDB – docs.mongodb.com | Stanford – online.stanford.edu
- India DPDP Act 2023 – meity.gov.in | FAISS – github.com/facebookresearch/faiss

**Recommended Free & Free-Tier AI / Database Tools:**

- Design & Modeling: draw.io; MySQL Workbench – ER diagrams and schema design
- SQL Databases: MySQL; SQLite (free, open-source) – query implementation and practice
- NoSQL & Vector: MongoDB Atlas; Redis; Neo4j; FAISS; Chroma – modern data stores
- Environment: Jupyter Notebook / Google Colab (free) – FD closure, query exercises
- AI Assistance: ChatGPT; Claude; GitHub Copilot – text-to-SQL, normalization, and verification

## B.Tech. – II Year II Semester

| Course Code | Category | Name of the Course | L | T | Pr | P | C |
|---|---|---|---|---|---|---|---|
| 26PC05206T | PC | Automata Theory and Compiler Design | 2 | 0 | 2 | 0 | 3 |

**Pre-Requisites:** Discrete Mathematics and basic programming in C / Python.

**Course Objectives:**

- To build a strong foundation in formal languages, automata, and the mathematical limits of computation.
- To analyze and construct finite automata, grammars, and Turing machines.
- To understand compiler design phases from lexical analysis to code generation.
- To design and evaluate parsing techniques and syntax-directed translation.
- To apply optimization techniques and understand target code generation.

**Course Outcomes (COs):**

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

**CO1:** Apply formal language hierarchy and automata models. (L3)

**CO2:** Construct and minimize DFA/NFA and derive regular expressions. (L3)

**CO3:** Analyze context-free grammars and design parsers (LL/LR). (L4)

**CO4:** Apply syntax-directed translation and semantic analysis. (L3)

**CO5:** Generate and optimize intermediate and target code. (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 | 1 | | 1 | | | | 2 | 3 | 2 |
| CO2 | 3 | 3 | 2 | 2 | 2 | | 1 | | | | 2 | 3 | 3 |
| CO3 | 3 | 3 | 3 | 2 | 2 | | 1 | | | | 2 | 3 | 3 |
| CO4 | 3 | 3 | 2 | 2 | 2 | | 1 | | | | 2 | 3 | 3 |
| CO5 | 3 | 3 | 3 | 3 | 2 | | 1 | | | | 2 | 3 | 3 |

**Unit I: Introduction to Finite Automata:**

**Core Content:** Introduction to Finite Automata: Alphabets, Strings, Languages. Finite Automata, Nondeterministic Finite Automata (NFA), Finite Automata with Epsilon-Transitions. Deterministic Finite Automata (DFA), String and Language Acceptance, Conversion of NFA with ε-transitions to NFA without ε-transitions, Conversion of NFA to DFA, DFA minimization, equivalence of DFA.

**AI Integration:** Stepwise DFA construction using ChatGPT; automata visualization explanations using Gemini; designing DFA and NFA for specific string acceptance; converting NFA with ε-transitions to NFA and NFA to DFA; DFA minimization and checking equivalence between machines.

**Suggested Tools:** JFLAP, Python

**Unit II: Regular Expressions & Context-Free Grammars:**

**Core Content:** Regular Expressions: Finite Automata and Regular Expressions, Applications of Regular Expressions, Algebraic Laws for Regular Expressions, Conversion of Finite Automata to Regular Expressions. Pumping Lemma for Regular Languages — statement of the pumping lemma, applications of the pumping lemma. Context-Free Grammars: definition of context-free grammars, derivations using a grammar, leftmost and rightmost derivations, the language of a grammar, parse trees, ambiguity in grammars and languages. Chomsky Normal Form, Greibach Normal Form.

**AI Integration:** Regex to DFA conversion; developing CFGs and parse trees while resolving ambiguity; grammar normalization (CNF, GNF); regex optimization via Claude AI; grammar correction using Gemini; parse tree generation.
**Suggested Tools:** JFLAP, draw.io

**Unit III: Pushdown Automata & Turing Machines:**
**Core Content:** Pushdown Automata: definition of the pushdown automaton, the languages of a PDA, equivalence of PDAs and CFGs, acceptance by final state and empty stack. Turing Machines: introduction to Turing machine, formal description, instantaneous description, the language of a Turing machine, types of Turing machine — Multi-Tape Turing Machine, Non-Deterministic Turing Machine.
**AI Integration:** Designing PDAs for acceptance by final state and empty stack; constructing Turing Machines and tracking Instantaneous Descriptions; building Multi-Tape and Non-Deterministic TM models; TM design logic and transition assistance via Claude AI; logic verification for complex machine models using ChatGPT.
**Suggested Tools:** JFLAP, Python

**Unit IV: Introduction to Compiler Design and Parsing:**
**Core Content:** Introduction to Compiler Design: the structure of a compiler. Lexical Analysis: the role of the lexical analyzer, input buffering, recognition of tokens. Syntax Analysis: introduction; Top-Down Parsing: Recursive Descent and LL(1); Bottom-Up Parsing: Shift-Reduce and Operator Precedence.
**AI Integration:** Implementing lexical analyzers for token recognition and input buffering; constructing LL(1) parsing tables and Recursive Descent parsers; applying Shift-Reduce and Operator Precedence bottom-up parsing; code generation for lexical analyzers using GitHub Copilot; parser debugging and error analysis via Perplexity AI.
**Suggested Tools:** Flex / Bison, VS Code

**Unit V: LR Parsing, Code Generation & Optimization:**
**Core Content:** Introduction to LR Parsing: Simple LR, CLR, LALR. Semantic Analysis: Syntax-Directed Translation Schemes. Intermediate-Code Generation: Three-Address Code. Code Optimization and Generation: principal sources of optimization, basic blocks and flow graphs, optimization of basic blocks, issues in the design of a code generator, the target language, a simple code generator, peephole optimization.
**AI Integration:** Optimization suggestions via Claude AI; code translation and research via NotebookLM; designing parsing tables for SLR, CLR, and LALR; generating Three-Address Code (TAC) and Syntax-Directed Translation; constructing flow graphs and applying peephole optimization.
**Suggested Tools:** GCC, LLVM, Graphviz

**Text Books:**

1. Michael Sipser, Introduction to the Theory of Computation, 3rd ed., Cengage Learning.
2. K. L. P. Mishra & N. Chandrasekaran, Automata Theory and Formal Languages, Revised ed., PHI Learning, 2022.
3. John C. Martin, Theory of Computation, 2021 ed., McGraw-Hill.

4. A. V. Aho, M. S. Lam, R. Sethi & J. D. Ullman, Compilers: Principles, Techniques and Tools, 2nd ed., Pearson.
5. K. V. Sunitha & N. Kalyani, Compiler Design, McGraw-Hill, 2021/2022.

**Web Resources:**

- NPTEL – Compiler Design & Theory of Computation – nptel.ac.in
- Stanford University – CS theory & compiler lecture materials – web.stanford.edu
- GeeksforGeeks – Automata theory & compiler design tutorials – geeksforgeeks.org
- LLVM Foundation – LLVM compiler infrastructure documentation – llvm.org/docs

**Recommended Tools & AI Assistants:**

- Operating System: Linux (Ubuntu).
- Programming: Python, C.
- Debugging: GDB, VS Code Debugger.
- Visualization: JFLAP, Graphviz, draw.io.
- Compiler Tools: Flex / Bison, GCC, LLVM.
- Containerization: Docker.
- AI Tools: ChatGPT, Claude, Gemini, Perplexity AI, GitHub Copilot, NotebookLM.

## B.Tech. – II Year II Semester

| Course Code | Category | Name of the Course | L | T | Pr | P | C |
|---|---|---|---|---|---|---|---|
| 26PC05207T | PC | Software Engineering for AI Driven Systems | 2 | 0 | 2 | 0 | 3 |

**Pre-Requisites:** Programming fundamentals and basic problem-solving skills.

**Course Objectives:**

- To understand software engineering principles, process models, and software lifecycle activities.
- To analyze software requirements and apply project management techniques for software development.
- To apply software design principles and agile engineering practices for building quality software systems.
- To evaluate software quality through testing methodologies and software engineering automation practices.
- To apply AI-assisted tools and techniques throughout the software development lifecycle.
- To understand software engineering practices for AI-enabled systems and emerging development environments.

**Course Outcomes (COs):**

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

**CO1:** Apply software engineering principles, lifecycle modelssoftware development practices. (L3)

**CO2:** Analyze software requirements and apply project estimation and management techniques. (L4)

**CO3:** Apply software design methodologies and agile practices to software systems. (L3)

**CO4:** Evaluate software quality using testing, reliability, and software engineering automation practices. (L5)

**CO5:** Design software solutions using AI-enabled tools and practices for modern software systems. (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 | 2 | 1 | 1 | 2 | | 1 | | | | 2 | 2 | 3 |
| CO2 | 3 | 3 | 2 | 2 | 2 | | 1 | | | | 2 | 3 | 3 |
| CO3 | 3 | 2 | 3 | 2 | 2 | | 1 | | | | 2 | 2 | 3 |
| CO4 | 3 | 3 | 2 | 3 | 2 | | 1 | | | | 2 | 2 | 3 |
| CO5 | 3 | 3 | 3 | 3 | 3 | | 1 | | | | 3 | 3 | 3 |

**Unit I: Software Engineering Foundations and Process Models:**

**Core Content:** Software Engineering Fundamentals: evolution of software systems, software crisis, exploratory style of development, software development practices, computer system engineering principles. Software Process Models: waterfall model and extensions, incremental model, Rapid Application Development (RAD), spiral model, agile development model; comparative analysis of process models. Software life cycle concepts; software development projects. Emerging trends: human–AI collaboration in software development; AI-assisted software lifecycle overview.

**AI Integration:** Use ChatGPT or Claude to generate software project scenarios and compare SDLC models; use Gamma.app or Notion AI to create process model explanations and lifecycle visualizations; use Miro or Lucidchart AI to visualize software process workflows.

**Suggested Tools:** ChatGPT, Claude, Gamma.app, Notion AI, Miro, Lucidchart AI, GitHub Copilot

**Unit II: Software Project Management and Requirements Engineering:**

**Core Content:** Software Project Management: complexities, responsibilities of project managers, metrics for project size estimation, project estimation techniques, empirical estimation techniques, COCOMO, Halstead software science, risk management. Requirements Engineering: requirements gathering, requirements analysis, Software Requirements Specification (SRS). Formal specification methods: axiomatic specification, algebraic specification, executable specification; Fourth Generation Languages (4GL). AI-Enabled Requirements Engineering: automated requirement extraction, AI-assisted documentation generation, requirement consistency analysis.

**AI Integration:** Use ChatGPT or Claude to generate and refine SRS documents; use Notion AI for requirement documentation and organization; use Jira AI or Linear for project planning and task management; use Claude to validate requirements and identify risks.

**Suggested Tools:** ChatGPT, Claude, Notion AI, Jira AI, Linear

**Unit III: Software Design and Agile Engineering Practices:**

**Core Content:** Software Design Principles: characteristics of good design, layered arrangement of modules, cohesion and coupling, software design approaches. Function-Oriented Software Design: Structured Analysis and Design, Data Flow Diagrams (DFD), structured design, detailed design, design review. Agile Engineering Practices: agility and cost of change, agile process models, Extreme Programming (XP), agile toolsets. User Interface Design: characteristics of user interfaces, GUI concepts, component-based GUI development, user interface methodology. Modern Development Approaches: component-based development, low-code / no-code application development, comparison with traditional development.

**AI Integration:** Use StarUML or Visual Paradigm AI for UML modeling; use Figma AI or Uizard for user interface prototyping; use Mendix or OutSystems for low-code development; use AppGyver or Bubble for no-code development; use Miro AI or draw.io for DFD and workflow creation.

**Suggested Tools:** StarUML, Visual Paradigm AI, Figma AI, Uizard, Mendix, OutSystems, Bubble, AppGyver, Miro AI, draw.io

**Unit IV: Software Testing, Quality Engineering, and Automation:**

**Core Content:** Software Testing: coding and code review, software documentation, black-box testing, white-box testing, debugging, program analysis tools, integration testing, testing object-oriented programs, smoke testing. Software Quality Engineering: software reliability, statistical testing, quality management systems — ISO 9000, Capability Maturity Model (CMM), Six Sigma. Software Engineering Automation Practices: automated test generation, continuous testing concepts, automated code review, static code analysis, AI-assisted defect prediction.

**AI Integration:** Use GitHub Copilot for code generation and AI-assisted code review; use Testim or Selenium AI for automated test case generation and testing; use SonarQube for static code analysis; use DeepCode or Snyk for AI-based defect identification.

**Suggested Tools:** GitHub Copilot, Claude, Testim, Selenium AI, SonarQube, DeepCode, Snyk, ChatGPT

**Unit V: Software Engineering for AI-Driven Systems:**

**Core Content:** AI System Engineering Fundamentals: characteristics of AI systems, traditional software vs. AI systems, data-centric development. AI Lifecycle Management: data pipelines, feature engineering, model lifecycle, model validation, model deployment. AI-Supported Software Maintenance: predictive maintenance using AI, AI-assisted bug localization, reverse engineering

using AI, AI-generated documentation, model drift and retraining. Responsible AI Engineering: explainable AI (XAI) concepts, bias and fairness, ethical considerations, AI governance, monitoring AI systems.

**AI Integration:** Use Google Colab or Jupyter Notebook for AI workflow experimentation; use MLflow for model tracking and lifecycle management; use Evidently AI or Grafana for model monitoring and drift visualization; use Hugging Face Hub to explore pretrained models; use Claude or Notion AI for AI-generated documentation and maintenance support.

**Suggested Tools:** Google Colab, Jupyter Notebook, MLflow, Evidently AI, Grafana, Hugging Face Hub, Claude, Notion AI

**Text Books:**

1. R. S. Pressman & B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed., McGraw-Hill Education, 2020.
2. I. Sommerville, Software Engineering, 10th ed., Pearson Education, 2016.
3. C. Huyen, AI Engineering: Building Applications with Foundation Models, O'Reilly Media, 2025.

**Reference Books:**

1. C. Huyen, Designing Machine Learning Systems, O'Reilly Media, 2022.
2. V. Lakshmanan, S. Robinson & M. Munn, Machine Learning Design Patterns, O'Reilly Media, 2020.
3. P. Jalote, An Integrated Approach to Software Engineering, 4th ed., Springer, 2017.
4. S. Russell & P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Pearson, 2020.

**Online Resources:**

- NPTEL – Software Engineering – nptel.ac.in
- MLflow Documentation – mlflow.org
- Evidently AI – evidentlyai.com
- Hugging Face – huggingface.co
- SonarQube – sonarqube.org
- GitHub Copilot – github.com/features/copilot
- IEEE Software Engineering Standards – ieee.org

**Recommended Free & Free-Tier AI / Software Engineering Tools:**

- AI Assistants: ChatGPT, Claude – scenario generation, SRS drafting, code review and documentation.
- Design & Modeling: StarUML, Visual Paradigm, Figma, draw.io, Miro – UML, UI prototyping and workflow diagrams.
- Project & Docs: Jira, Linear, Notion AI, Gamma.app – planning, estimation and documentation.
- Testing & Quality: Testim, Selenium, SonarQube, Snyk, DeepCode – automated testing, static analysis and defect prediction.
- AI Lifecycle: Google Colab, Jupyter, MLflow, Evidently AI, Hugging Face Hub – experimentation, model tracking and monitoring.
- Low-Code / No-Code: Mendix, OutSystems, Bubble, AppGyver – rapid application development.

## B.Tech. – II Year II Semester

| Course Code | Category | Name of the Course | L | T | Pr | P | C |
|---|---|---|---|---|---|---|---|
| 26PC05204P | PC | Operating Systems and Intelligent Resource Management Lab | 0 | 0 | 0 | 3 | 1.5 |

**Pre-Requisites:** Programming in C and Python; basic Linux familiarity.

**Course Objectives:**

- To develop practical understanding of the Linux environment, process management, memory management, file systems, and system resource utilization.
- To apply scheduling, synchronization, deadlock handling, virtual memory, and disk management concepts.
- To design concurrent and resource-management solutions using system calls and multithreading.
- To integrate AI techniques for system monitoring and intelligent resource optimization.
- To gain exposure to containerization and intelligent deployment environments.

**Course Outcomes (COs):**

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

**CO1:** Demonstrate Linux environment usage and process management using system calls. (L3)

**CO2:** Apply CPU scheduling, synchronization, and multithreading concepts. (L3)

**CO3:** Implement memory management, virtual memory, and file-system operations. (L3)

**CO4:** Analyse disk scheduling and resource allocation mechanisms. (L4)

**CO5:** Develop intelligent OS applications using AI and container technologies.. (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 | 2 | 2 | 3 | | 1 | | | | 2 | 3 | 3 |
| CO2 | 3 | 3 | 2 | 2 | 3 | | 1 | | | | 2 | 3 | 3 |
| CO3 | 3 | 3 | 2 | 2 | 3 | | 1 | | | | 2 | 3 | 3 |
| CO4 | 3 | 3 | 2 | 3 | 3 | | 1 | | | | 2 | 3 | 3 |
| CO5 | 3 | 3 | 3 | 3 | 3 | | 1 | | | | 3 | 3 | 3 |

**List of Experiments**

**Module I: Linux Environment and Process Management**

**Topics:** Linux installation and environment; file, directory, permission, and process commands; process creation using fork(), exec(), and wait(); process trees and parent/child memory comparison.

**AI Integration:** Explain Linux command functionality and outputs; generate practice exercises; explain process-creation sequence; generate process trees; debug process-creation errors.

**Suggested Tools:** Linux (Ubuntu), C (GCC), ChatGPT, Microsoft Copilot, GitHub Copilot

**Experiments:**

- Experiment 1: Linux Commands and Environment — Install Ubuntu/Linux (physical, virtual, or cloud); execute file, directory, permission, and user/process management commands; create a project directory structure, modify permissions, and monitor running processes with CPU and memory utilization.
- Experiment 2: Process Creation using System Calls — Write a C program using fork() to create a child process; display PID and PPID; use exec() and wait(); observe execution order, compare parent/child memory before and after exec(), and draw the process tree.

**Module II: CPU Scheduling, Synchronization & Multithreading**

**Topics:** CPU scheduling — FCFS, SJF, Priority, Round Robin; Gantt charts and performance metrics; multithreading; producer–consumer, readers–writers, and dining-philosophers synchronization.

**AI Integration:** Generate process datasets and scenarios; verify scheduling calculations; generate Gantt charts; compare algorithm performance; detect race conditions and suggest synchronization improvements; generate synchronization traces.

**Suggested Tools:** C / Python, pthreads, semaphores / mutex, ChatGPT, GitHub Copilot

**Experiments:**

- Experiment 3: Process Scheduling – FCFS and SJF — Read process information; implement FCFS and non-preemptive SJF scheduling; display Gantt charts, completion/waiting/turnaround times and averages; compare the two on average waiting time, turnaround time, and CPU utilization with a comparison graph.
- Experiment 4: Priority & Round Robin Scheduling — Implement Priority and Round Robin scheduling; compare fairness and efficiency across scenarios.
- Experiment 5: Multithreading — Create threads using pthreads/Python; execute concurrent tasks; compare sequential and parallel execution.
- Experiment 6: Producer–Consumer Synchronization — Implement producer–consumer using semaphores/mutex with a shared buffer; detect and resolve race conditions.
- Experiment 7: Readers–Writers / Dining Philosophers — Implement a classic synchronization problem while avoiding deadlock and starvation; analyse resource contention.

**Module III: Memory Management and File Systems**

**Topics:** Memory allocation — First Fit, Best Fit, Worst Fit; page replacement — FIFO, LRU, Optimal; Banker's Algorithm; file-system and directory operations.

**AI Integration:** Visualize allocation and compare utilization; simulate page replacement and compare efficiency; generate allocation scenarios and validate deadlock prevention; generate file structures and validate operations.

**Suggested Tools:** C / Python, Jupyter Notebook, Google Colab, ChatGPT, GitHub Copilot

**Experiments:**

- Experiment 8: Memory Allocation Techniques — Implement First Fit, Best Fit, and Worst Fit; compare fragmentation.
- Experiment 9: Page Replacement Algorithms — Implement FIFO, LRU, and Optimal; compute page faults and compare efficiency.
- Experiment 10: Banker's Algorithm — Implement Banker's Algorithm; identify safe and unsafe states; validate deadlock prevention.
- Experiment 11: File System Operations — Create, read, and write files; perform directory operations; simulate file allocation.

**Module IV: Disk Management and System Analytics**

**Topics:** Disk scheduling — FCFS, SSTF, SCAN, C-SCAN; system monitoring and log analysis.

**AI Integration:** Generate disk requests and compare performance; detect anomalies and interpret system behaviour.

**Suggested Tools:** C / Python, Jupyter Notebook, ChatGPT

**Experiments:**

- Experiment 12: Disk Scheduling Algorithms — Implement FCFS, SSTF, SCAN, and C-SCAN; compare seek time.
- Experiment 13: System Monitoring & Log Analysis — Monitor CPU and memory; analyse system logs; detect anomalies and interpret behaviour.

**Module V: Intelligent Resource Management**

**Topics:** Workload prediction using machine learning; containerization with Docker.

**AI Integration:** Generate visualizations and compare predictions; generate Dockerfiles and explain deployment.

**Suggested Tools:** Python, Scikit-learn, Google Colab, Docker, ChatGPT

**Experiments:**

- Experiment 14: Workload Prediction using ML — Collect CPU-usage data; build a prediction model; predict workload and compare predictions.
- Experiment 15: Docker Containerization — Install Docker; create and deploy a container; generate a Dockerfile and explain deployment.

**Reference Books:**

1. A. Silberschatz, P. B. Galvin & G. Gagne, Operating System Concepts.
2. A. S. Tanenbaum & H. Bos, Modern Operating Systems.
3. W. Stallings, Operating Systems: Internals and Design Principles.
4. Sumitabha Das, UNIX Concepts and Applications.
5. Robert Love, Linux System Programming.
6. Brendan Gregg, Systems Performance.
7. Nigel Poulton, Docker Deep Dive.
8. Aurélien Géron, Hands-On Machine Learning.

**Online Resources:**

- Linux Documentation Project – tldp.org
- https://www.youtube.com/@JoyofLearningCSE/featured
- Docker Documentation – docs.docker.com
- Scikit-learn Documentation – scikit-learn.org
- Jupyter Documentation – docs.jupyter.org
- NPTEL – Operating Systems – nptel.ac.in
- YouTube – Joy of Learning CSE – youtube.com/@JoyofLearningCSE

**Recommended Free & Open-Source OS / AI Tools:**

- Operating System: Linux (Ubuntu), VirtualBox / VMware – environment and process experimentation.
- Programming: C (GCC), Python, pthreads – system programming and concurrency.
- Notebooks & ML: Jupyter Notebook, Google Colab, Scikit-learn – simulation and workload prediction.
- Containers: Docker – containerization and deployment.
- AI Assistants: ChatGPT, GitHub Copilot, Microsoft Copilot – code generation, debugging and explanation.

## 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(AI&ML)) | 0 | 0 | 0 | 3 | 1.5 |

**Pre-Requisites:** Data Structures

**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 | 3 | 3 |
| CO2 | 3 | 2 | 3 | 2 | 3 | | 3 | 3 | 3 | | 2 | 3 | 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**

**Unit 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.

**Unit 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.

**Unit 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.

**Unit 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.

**Unit 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 Resources**

- NPTEL – Database Management System – nptel.ac.in
- MongoDB University – Free NoSQL Courses – university.mongodb.com
- Neo4j GraphAcademy – Free Graph Database Courses – graphacademy.neo4j.com

**Recommended Free & Open-Source AI / Computational Tools**

- AI Assistants: ChatGPT; GitHub Copilot – query generation, debugging, and schema/normalization validation.
- Modelling Tools: draw.io – ER diagram design.
- Databases & Engines: MongoDB (Atlas free tier); Neo4j (Community/AuraDB Free) – NoSQL and graph database experimentation.
- Vector Search Libraries: FAISS; Chroma – similarity search and vector database applications.
- Frameworks: LangChain – building Text-to-SQL and RAG-over-SQL pipelines.
- 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(AI&ML)) | 1 | 0 | 0 | 2 | 2 |

**Pre-Requisites:** Basic knowledge of Python programming and programming fundamentals.

**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 | 3 | 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**

**Unit I: Data Handling Using Python:**

**Topics:** Data Analytics Workflow; Jupyter Notebook Environment; Data Sources and File Formats; Importing Data; CSV, Excel and JSON Processing; NumPy Arrays; Data Manipulation; Vectorized Operations; Data Export.

**AI Integration:** Dataset understanding, automatic profiling and analytical assistance.

**Suggested Tools:** Python, Jupyter Notebook, Pandas, NumPy

**Experiments:**

- Experiment 1: Dataset Import and Exploration — Import datasets from CSV, Excel and JSON formats. Examine dimensions, data types, missing values and generate summary statistics.
- Experiment 2: Numerical Data Processing using NumPy — Perform array creation, indexing, slicing, aggregation, mathematical operations and vectorized computations.

- Experiment 3: DataFrame Operations using Pandas — Create DataFrames and perform selection, filtering, indexing, sorting and aggregation.

**Unit II: Data Processing and Transformation Using Pandas:**
**Topics:** Series and DataFrames; Data Selection; Indexing and Filtering; Missing Data Handling; Data Cleaning; Aggregation; Grouping; Merging; Sorting; Feature Construction (without ML).
**AI Integration:** AI-assisted cleaning, transformation recommendations and optimization.
**Suggested Tools:** Pandas, Jupyter Notebook
**Experiments:**

- Experiment 4: Data Cleaning and Missing Value Handling — Identify duplicate records, missing values and outliers. Apply cleaning and preprocessing techniques.
- Experiment 5: Data Transformation and Feature Construction — Perform grouping, merging, aggregation, reshaping and derived column generation.
- Experiment 6: Descriptive Statistics and Data Analysis — Generate statistical summaries and perform exploratory analysis including Mean, Median, Mode, Variance and Distribution Analysis.

**Unit III: Exploratory Data Analysis and Visualization:**
**Topics:** Descriptive Statistics; Data Exploration; Correlation Analysis; Data Visualization Principles; Line Charts; Bar Charts; Scatter Plots; Histograms; Boxplots; Heatmaps; Interactive Visualization.
**AI Integration:** Visualization recommendation, dashboard generation and insight generation.
**Suggested Tools:** Matplotlib, Seaborn, Plotly
**Experiments:**

- Experiment 7: Data Visualization using Matplotlib — Generate visual reports using line charts, bar charts, histograms and pie charts.
- Experiment 8: Advanced Visualization using Seaborn — Develop scatter plots, boxplots, heatmaps and distribution analysis.
- Experiment 9: Interactive Visualization and Dashboard Design — Build interactive visualizations and analytical dashboards.
- Experiment 10: Correlation and Comparative Data Analysis — Perform comparative analysis and identify relationships among variables.

**Unit IV: Data Analytics Using Real-World Datasets:**
**Topics:** Data Collection Methods; Public Dataset Usage; JSON Processing; API-based Data Collection; Time-Series Data Handling; Data Transformation; Report Generation; Dashboard Concepts.
**AI Integration:** Automated analytics, report generation and insight extraction.
**Suggested Tools:** Pandas, Requests, Plotly
**Experiments:**

- Experiment 11: Data Collection using APIs and JSON Processing — Collect data using APIs and process JSON data.
- Experiment 12: Time-Series and Trend Analysis — Analyze temporal datasets and generate trend reports including date handling, rolling statistics and visualization.
- Experiment 13: Analytical Report Generation — Generate analytical summaries and reports from datasets.

**Unit V: Applied Data Analytics Project:**

**Topics:** Analytics Workflow; Data Preparation; Data Visualization; Insight Generation; Dashboard Development; Documentation; Project Presentation.

**AI Integration:** Data profiling, visualization recommendation, reporting and documentation.

**Suggested Tools:** Python, Jupyter Notebook, Plotly

**Experiments:**

- Experiment 14: Dashboard Development using Real-world Datasets — Create complete dashboards for decision support using Education, Healthcare, Retail or Weather datasets.
- Experiment 15: Applied Data Analytics Mini Project — Develop and present a complete analytics solution including data collection, data cleaning, analysis, visualization, dashboard and report generation.

**Suggested Mini Projects:**

1. Student Performance Analytics
2. Retail Sales Analytics
3. Healthcare Analytics
4. Weather Analytics
5. Financial Data Analytics
6. Social Media Analytics
7. Sports Analytics
8. Survey Data Analytics

**Text Books:**

1. Wes McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy and Jupyter, O'Reilly Media.
2. Jake VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
3. Joel Grus, Data Science from Scratch: First Principles with Python, O'Reilly Media.

**Reference Books:**

1. Cathy O'Neil and Rachel Schutt, Doing Data Science, O'Reilly Media.
2. Trevor Hastie, Robert Tibshirani and Gareth James, An Introduction to Statistical Learning, Springer.
3. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Media.
4. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow, O'Reilly Media (selected chapters on data preparation and analysis).
5. Ethem Alpaydin, Introduction to Machine Learning, MIT Press (reference for analytics foundation).

**Online Learning Resources:**

- Python Documentation – docs.python.org
- Jupyter Documentation – docs.jupyter.org
- NumPy Documentation – numpy.org/doc
- Pandas Documentation – pandas.pydata.org/docs
- Matplotlib Documentation – matplotlib.org/stable
- Seaborn Documentation – seaborn.pydata.org
- Plotly Documentation – plotly.com/python

**Recommended Free & Open-Source AI / Computational Tools:**

- Core Libraries: Python, Pandas, NumPy – data handling, manipulation and numerical computing.
- Visualization: Matplotlib, Seaborn, Plotly – static and interactive charts and dashboards.
- Environment: Jupyter Notebook – interactive analysis, experimentation and reporting.
- Data Access: Requests – API-based data collection and JSON processing.
- AI Assistants: ChatGPT, Claude, GitHub Copilot – code generation, automatic data profiling and analytical assistance.

## B.Tech. – II Year II 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:** Apply the principles, types, and structure of technical reports, and explain the fundamentals of Intellectual Property Rights. (L3)

**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 | 2 | 1 |
| CO2 | 2 | 2 | 2 | 2 | | | 2 | 2 | 3 | 2 | 3 | 2 | 1 |
| CO3 | 2 | 3 | 2 | 2 | | | 2 | 2 | 3 | 2 | 3 | 2 | 2 |
| CO4 | 2 | 3 | 2 | 3 | | | 3 | 2 | 2 | 2 | 3 | 2 | 2 |
| CO5 | 2 | 2 | 3 | 2 | | | 2 | 3 | 3 | 3 | 3 | 2 | 2 |

**Unit I: Foundations of Technical Communication:**

**Core 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:**

**Core 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 generaon. AI Tool: ChatGPT for first-draft abstract generation. Plagiarism check using Turnitin or Copyleaks.

**Unit III: Research Proposals and Scientific Writing:**

**Core 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:**

**Core 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:**

**Core 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