NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY

DEEMED TO BE UNIVERSITY UNDER DISTINCT CATEGORY

(AN AUTONOMOUS INSTITUTION UNDER MINISTRY OF ELECTRONICS & IT, GOVT. OF INDIA)

Curriculum and Syllabi for

B.Tech - Computer Science and Engineering

(2026-27 Scheme)

Main Campus at Ropar and Constituent Units at Aizawl, Agartala, Aurangabad, Calicut, Gorakhpur, Imphal, Itanagar, Kekri, Kohima, Patna and Srinagar

Vision

To be known globally for value-based education, research, and innovation through academic excellence to serve the needs of industry and society.

Mission

1. Develop a strong foundation on fundamentals of engineering through outcome-based teaching and learning process.
2. Provide opportunities for students to work on real-world issues and develop sustainable ethical solutions.
3. Involve the students in group activities, including those of professional bodies, to develop leadership, entrepreneurship, and good communication skills.

Program Education Objectives

PEO1: Graduates will demonstrate their professional knowledge in their area of specialization and allied fields.
PEO2: Graduates will contribute to research and innovation with the design and development of innovative products using emerging technologies.
PEO3: Graduates will become successful leaders and entrepreneurs through effective project management and contribute towards the growth & development of industry & society.
PEO4: Graduates will be involved in promoting professional and societal activities.

Program Outcomes

PO1 - Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2 - Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems, reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3 - Design and Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4 - Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5 - Modern Tools Usage: Create, select, and apply appropriate techniques, resources, and modern engineering & IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6 - The Engineer and Society: Apply reasoning informed by the contextual knowledge to access societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.
PO7 - Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8 - Ethics: Apply ethical principles and commit to professional ethics, responsibilities, and norms of the engineering practice.
PO9 - Individual and Teamwork: Function effectively as an individual, and as a member of leader in diverse terms, and in multidisciplinary settings.
PO10 - Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give & receive clear instructions.
PO11 - Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12 - Life-long Learning: Recognize the need and have the preparation & ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

PSO1: Graduates should be able to understand the concepts of Computer Science and Engineering, with their applications in sub-fields and interdisciplinary research.
PSO2: Graduates should have an ability to apply technical knowledge and usage of modern hardware and software tools related to Computer Science and Engineering for solving real-world issues.
PSO3: Graduates should have the capability to analyze, comprehend, design, and develop software systems for a variety of engineering applications, demonstrating professional ethics and concern for societal well-being.

CATEGORY WISE CREDIT DISTRIBUTION

Category	Category Code	Credits
Open Elective Courses	OE	12
Ability Enhancement Courses	AE	8
Skill / Employment Enhancement Courses (Project/Internship/Seminar/Dissertation/Research)	EE	31
Program Core Courses	PC	88
Audit Courses (without grade or credit)	AU	0
Value Added Courses	VA	7
Professional Elective Courses	PE	20
Total Credits (Common track)		**166**

EVALUATION AND ASSESSMENT

1. Performance of a student in a semester shall be evaluated through continuous Class assessment, Tutorial/Lab assessment, Mid-Semester Examination (MSE) and End-Semester Examination (ESE). Both the MSE and ESE shall be the University examination and will be conducted as notified by the Controller of Examinations (CoE) of the University.

2. The continuous assessment shall be based on assignments, tutorials, paper presentation / quizzes/ viva-voce / flipped classes, lab work / projects / fieldwork and attendance, etc.

3. The MSE/ESE shall be comprising of written papers.

4. The overall assessment of the students will be done as per the following scheme:

S. No.	Assessment Type	Marks Weightage
1.	Mid Semester Examination	20
2.	End Semester Examination	40
3.	Continuous Assessment - Practical /Lab/ Tutorial	25
4.	Continuous Assessment - Theory	15
Total		**100**

For more details, please refer the applicable ordinance for this programme and Assessment SOPs.

CURRICULUM

Semester 1						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-101	DASH250038	Engineering Mathematics I	3	1	0	4
EE-BTCCSE-102	DCSE250039	Programming Fundamentals	2	0	4	4
PC-BTCCSE-103	DOEE250033	Basic Electrical & Electronics Engineering	2	1	2	4
PC-BTCCSE-104	DASH250042	Engineering Physics	3	0	2	4
AE-BTCCSE-105	DASH250016	Communication Skills	1	0	2	2
OE-BTCCSE-106	Elective	Open Elective				4

Semester 2						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-201	DASH250039	Engineering Mathematics II	3	1	0	4
PE-BTCCSE-202	Elective	Program Elective				4
PC-BTCCSE-203	DOEE250059	Digital Electronics	3	0	2	4
EE-BTCCSE-204	DCSA250016	Computer Aided Graphics Design	1	1	4	4
VA-BTCCSE-205	DASH250030	Economics for Engineers	1	1	0	2
OE-BTCCSE-206	Elective	Open Elective				4

Semester 3						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-301	DASH250028	Discrete Mathematics	3	1	0	4
PC-BTCCSE-302	DCSE250055	Data Structures	3	0	2	4
PC-BTCCSE-303	DOEE250044	Computer Organization and Architecture	3	0	2	4
PC-BTCCSE-304	DOEE250022	Assembly Language and Microprocessors	3	0	2	4
AE-BTCCSE-305	DASH250104	Technical Presentation and Report Writing	0	0	4	2
PE-BTCCSE-306	Elective	Program Elective				4
AU-BTCCSE-307	Elective	Audit Elective				0

Semester 4						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-401	DCSE250120	Theory of Computation	3	1	0	4
PC-BTCCSE-402	DCSE250010	Design and Analysis of Algorithms	3	0	2	4
PC-BTCCSE-403	DCSE250102	Operating Systems	3	0	2	4
VA-BTCCSE-404	DASH250052	Environmental Science	2	0	0	2
OE-BTCCSE-405	Elective	Open Elective				4
EE-BTCCSE-406	EE	Research and Survey Report				4

Semester 5						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-501	DCSE250107	Principles of Compiler Design	3	0	2	4
PC-BTCCSE-502	DCSE250060	Database Management Systems	2	0	4	4
PC-BTCCSE-503	DCSE250035	Computer Networks	3	0	2	4
AE-BTCCSE-504	DASH250026	Design Thinking	1	0	2	2
PE-BTCCSE-505	Elective	Program Elective				4
EE-BTCCSE-506	EE	Internship				4

Semester 6						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-601	DCSA250056	Numerical Methods	3	1	0	4
PC-BTCCSE-602	DCSE250082	Game Theory and Mechanism Design	3	1	0	4
PC-BTCCSE-603	DCSE250043	Cryptography and Network Security	3	0	2	4
AE-BTCCSE-604	DASH250056	Financial Accounting & Management	2	0	0	2
VA-BTCCSE-605	DASH250105	Universal Human Values-II: Understanding Harmony And Ethical Human Conduct	3	0	0	3
PE-BTCCSE-606	Elective	Program Elective				4

Semester 7						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
PC-BTCCSE-701	DCSE250004	Advanced Data Structures and Algorithms	3	0	2	4
PC-BTCCSE-702	DCSA250071	Software Engineering	3	1	0	4
PC-BTCCSE-703	DCSE250073	Ethical Hacking	2	1	2	4
PC-BTCCSE-704	DCSA250031	Full Stack Web Development	3	0	2	4
PE-BTCCSE-705	Elective	Program Elective				4
EE-BTCCSE-706	EE	Mini Project				5

Semester 8						
Semester Code	**Course Code**	**Course Title**	**L**	**T**	**P**	**Cr**
EE-BTCCSE-801	EE	Project and Thesis				10

Elective Courses

Elective Courses for PE Category						
Course Code	**Course Title**	**L**	**T**	**P**	**Cr**	**For Sem./Code**
DOAI260001	AI (Industry)-Applied Ethics for AI	3	0	2	4	
DOAI260002	AI (Industry)-Introduction to Artificial Intelligence	3	0	2	4	
DOAI260003	AI (Industry)-Introduction to Generative AI	3	0	2	4	
DOAI260004	AI (Industry)-Introduction to Machine Learning	3	0	2	4	
DCSA250057	Object Oriented Programming using C++	3	0	2	4	202
DCSA250059	Object-Oriented Programming with Java	2	0	4	4	202
DCSE250113	Programming in Python	2	0	4	4	306
DOEE250078	Electronic Devices and Circuits	3	0	2	4	306
DOAI250005	Artificial Intelligence and Machine Learning	2	0	4	4	505
DOAI250021	Data Science	3	0	2	4	505
DCSE250040	Concurrent and Parallel Processing	3	0	2	4	606
DCSE250068	Distributed Computing and Networking	3	0	2	4	606
DCSA250038	Introduction to Blockchain Technology	2	1	2	4	705
DCSE250020	Cloud Computing	3	0	2	4	705

Elective Courses for OE Category
Students may opt courses under MOOC, NPTEL, SWAYAM, NSQF/NCVET aligned courses or other relevant technical subjects not included in their core curriculum. The exercise of option shall be subject to the Course Schedule of the respective Constituent Unit, the credit requirements and availability of seats. Guidelines for choosing MOOC/Online courses are given separately and shall have to be adhered to.

Elective Courses for AU (Audit) Category					
Course Code	**Course Title**	**L**	**T**	**P**	**Cr**
DASH240027	Disaster Management	2	0	0	0
DASH240047	English for Research Paper Writing	2	0	0	0
DASH240052	Environmental Science	3	0	0	0
DASH240085	Pedagogy Studies	2	0	0	0
DASH240086	Personality Development through Life Enlightenment Skills	2	0	0	0
DASH240097	Sanskrit for Technical Knowledge	2	0	0	0
DASH240103	Stress Management by Yoga	2	0	0	0
DASH240104	Technical Presentation and Report Writing	0	0	2	0
DASH240106	Value Education	2	0	0	0
DASH240124	Constitution of India - AU	2	0	0	0
DASH250023	Constitution of India	2	1	0	0
DASH250027	Disaster Management	3	0	0	0
DASH250034	Energy and Environmental Engineering	3	0	0	0
DASH250047	English for Research Paper Writing	2	0	0	0

DASH250054	Essence of Indian Knowledge and Tradition	2	0	0	0
DASH250085	Pedagogy Studies	2	0	0	0
DASH250086	Personality Development	2	0	0	0
DASH250097	Sanskrit for Technical Knowledge	3	0	0	0
DASH250103	Stress Management by Yoga	3	0	0	0
DASH250106	Value Education	2	0	0	0
DASH250117	Innovative Thinking and Entrepreneurship Skills	1	0	0	0
DASH250118	Intellectual Property Rights	2	0	0	0
DCSA240080	Training on Computer Networking	0	0	2	0
DCSA240304	Lab Based on Statistical Package	0	0	2	0
Any other Scheduled Course as per the Course Schedule of the respective Constituent Unit can also be chosen, subject to availability of seats. However, there shall be no assessment and grading for the course chosen as Audit Course.					

The choice of all type of Electives available shall be as per the Course Schedule of the respective Constituent Unit.

Guidelines for Massive Open Online Courses (MOOC) / approved Online Courses

1. Relevant Swayam, MOOC, NPTEL, NIELIT NSQF and any other online courses approved by UGC/AICTE shall also be allowed as Open Electives(OE).

2. One credit of MOOC course should be minimum of 30 hours.

3. A student can choose any approved online course as Open Elective, subject to minimum credit requirements.

4. The students willing to opt OE under MOOC in their next semester, will submit their request to the MOOC Coordinator at their respective campus.

5. Once approved, the campus shall display the list of accepted courses.

6. The student has to submit certificate issued by the institution offering the MOOCs course, along with the number of credits and grades, to get credits transferred into his/her marks certificate issued by NDU.

7. The transfer of credits shall be as adopted by NDU.

Detailed Syllabus

Course Code	Course Name	L	T	P	Cr
DASH250038	Engineering Mathematics I	3	1	0	4

Course Outcomes:

CO1 : Apply matrix operations to solve systems of linear equations, find eigenvalues/eigenvectors, and use the Cayley-Hamilton theorem.
CO2 : Analyze functions of a single variable for limit, continuity and differentiability, Mean value theorems, definite integrals in engineering applications such as areas and volumes.
CO3 : Solve first and second-order ordinary differential equations, including linear, exact, and simultaneous forms, and interpret their geometric and analytical solutions.
CO4 : Differentiate multivariable functions, use Euler's theorem, Jacobians, and Taylor series, and solve problems involving maxima and minima of functions of two variables.
CO5 : Evaluate double and triple integrals in different coordinate systems and apply them to compute areas and volumes, including changing the order of integration when required.

Module 1:

Matrices (9 hrs)
Rank of a matrix, Consistency of a system of linear equations, Characteristic equation of a matrix, Eigen values and Eigen vectors, Cayley-Hamilton theorem (excluding proof), Verification, Application (Finding Inverse of a matrix), Diagonalization of a matrix by orthogonal transformation.

Module 2:

Differential Calculus and Applications (9 hrs)
Limit, Continuity and differentiability of a function of single variable, Rolle's theorem, Lagrange's theorem, and Cauchy's mean value theorem (Statements and simple applications), Applications of definite integrals to evaluate surface areas and volumes of revolutions of curves in Cartesian coordinates.

Module 3:

Differential Equations (7 hrs)
Ordinary Differential Equations (ODE): First order differential equations, Exact differential equations and integrating factor, Linear differential equations of first and second order, Simultaneous linear differential equations by elimination method.

Module 4:

Multivariable Calculus (Partial Differentiation and Applications) (11 hrs)
Partial differentiation, Partial derivatives of first, second and third order, Partial differentiation of implicit functions, Euler's theorem, Total derivative, Jacobian and its Properties, Applications: Maxima and minima of functions of two variables.

Module 5:

Multivariate Calculus (Multiple Integrals) (9 hrs)
Double integral (Cartesian, polar form), Application of double integral: Area by double integration, Change of Order of Integration. Triple Integral in Cartesian coordinate only, Application of triple integral: Volume by triple integration.

Labs / Practicals:

N/A

References:

1. Grewal, B. S. Higher Engineering Mathematics (43rd ed.). Khanna Publishers.
2. Bali, N. P., & Goyal, M. A Text Book of Engineering Mathematics. Laxmi Publications (P) Ltd.
3. Kreyszig, E. Advanced Engineering Mathematics (9th ed.). John Wiley & Sons.
4. Ramana, B. V. Higher Engineering Mathematics. Tata McGraw-Hill.
5. Jeffery, A. Advanced Engineering Mathematics. Academic Press.

Course Code	Course Name	L	T	P	Cr
DCSE250039	Programming Fundamentals	2	0	4	4

Course Outcomes:

CO1: Remember the syntax and basic operators in C language and its descendants.

CO2: Understand how to represent algorithms using flowcharts and pseudo-codes.

CO3: Apply well-known fundamental programming techniques for problem solving.

CO4: Analyze programming bugs through testing, stack traces, and core dumps.

CO5: Evaluate the outcomes of programs manually with dry runs and trace tables.

CO6: Create bug-free and efficient programming solutions for real-world problems.

Module 1:

Fundamentals of programming, program vs algorithm, implementing algorithms.
Language-independent algorithm representation using flowcharts and pseudo-codes.
Functionality vs implementation, usage of right abstractions, code documentation.
Runtime error vs logical bug, black box vs white box testing, debugging techniques.

Module 2:

Components of a program: constants, identifiers, operators, keywords, comments.
Editor vs compiler, compilation vs interpretation, basic I/O using printf and scanf.
Variable declaration vs definition, initialization, assignment, type promotion rules.
Arithmetic, relational, logical, bitwise operators, arity, precedence and associativity.

Module 3:

Sequential control flow, expression, statement, block, identifier scope, object lifetime.
Conditional operator, selection statements: if, else, switch; nested conditions.
Iteration blocks: for, while, do-while; loop invariants, nested loops, Duff's device.
Labels, jump statements: break, continue, goto; structured programming.

Module 4:

Function declaration vs definition, parameter vs argument, call by value, void type.
Local vs external variables, static variables, recursion, tail call, inline definition.
Arrays, random access, string, array vs pointer, array of array, variable-length array.
Enums, structures, bit-fields, padding, unions, compound literals, incomplete types.

Module 5:

Stack and heap layout, dynamic memory allocation, dangling pointer, memory leak.
Headers, preprocessing directives, multiple source files, internal vs external linkage.
Command-line arguments, C standard library, file I/O, buffering, signal handling.
Portability issues, undefined behavior, buffer overflow and other vulnerabilities.

Labs / Practicals:

All programming solutions should conform to ISO C99 standard (or later revisions).

01. Test the precedence and associativity of arithmetic operators in C: *, /, %, +, -.

02. Test the outcomes of relational and equality operators in C.

03. Test the short-circuit evaluation of && and || using function calls as operands.

04. Test the bitwise operators in C with unsigned and signed types: left shift, right shift, &, ^, |.

05. Convert from metric prefix (KB/MB/GB/TB) to binary prefix (KiB/MiB/GiB/TiB).

06. Use if-else-if ladder to print the grade as per the score obtained in a subject.

07. Find the mean, median, mode, variance, and standard deviation for a list of scores.

08. Print values using additive/subtractive rules of roman numerals: I, V, X, L, C, D, M.

09. Use nested loops to generate asymmetric and symmetric patterns of arbitrary size.

10. Check whether a given string is a palindrome (or not) without reversing the string.

11. Write a function to check whether an array of integers is already sorted (or not).

12. Write a function for primality testing, and use it to find first n pairs of twin primes.

13. Write a function to generate the Collatz sequence (up to 1) for a positive integer.

14. Generate the terms of Fibonacci sequence using iterative and recursive approaches.

15. Print optimal sequence of moves for a given instance of “Towers of Hanoi” puzzle.

16. Find all combinations to make a given payment from a limited set of coin types.

17. Find vowel count in a string using a lookup table for case-insensitive vowel check.

18. Write a function to check whether a string is a valid email ID (or not).

19. Use case-insensitive and dot-insensitive matching to check if two email IDs are same.

20. Display a countdown timer in HH:MM:SS format using carriage return ("\r").

21. Get current time and find clockwise angles of hour hand, minute hand, second hand.

22. Find the day of week for any date in Gregorian calendar using doomsday algorithm.

23. Populate an array with student details and write the attributes in a new CSV file.

24. Sort an array of student details as per roll number using qsort from stdlib.h .

25. Use bsearch from stdlib.h to fetch student details from an array sorted by roll.

References:

1. "Computer Science I" by Chris Bourke
https://cse.unl.edu/~cbourke/ComputerScienceOne.pdf

2. "How to Think Like a Computer Scientist (C Version)" by Thomas Scheffler and Allen Benjamin Downey
https://github.com/tscheffl/ThinkC/raw/master/PDF/Think-C.pdf

3. "Foundations of Computer Science (C Edition)" by Alfred Vaino Aho and Jeffrey David Ullman
http://infolab.stanford.edu/~ullman/focs.html

4. "The C Programming Language (Second Edition)" by Brian Wilson Kernighan and Dennis MacAlistair Ritchie
https://www.oreilly.com/library/view/c-programming-language/9780133086249

5. "C Programming: A Modern Approach (Second Edition)" by Kim N. King
http://knking.com/books/c2

6. "Modern C (Third Edition)" by Jens Gustedt
https://inria.hal.science/hal-02383654v2/file/modernC.pdf

7. "The C Book (Second Edition)" by Mike Banahan, Declan Brady, and Mark Doran
https://publications.gbdirect.co.uk/c_book

8. "Beej's Guide to C Programming" by Brian Hall
https://beej.us/guide/bgc/pdf/bgc_a4_c_2.pdf

Course Code	Course Name	L	T	P	Cr
DOEE250033	Basic Electrical & Electronics Engineering	2	1	2	4

Course Outcomes:

CO1: Analyze basic DC and AC electrical circuits using Ohm's Law, Kirchhoff's Laws, and network theorems.
CO2: Understand and explain power generation methods, transmission, and distribution, and describe the components of an electrical power system.
CO3: Identify and describe the construction and working principles of transformers, DC machines, and induction motors.
CO4: Analyze semiconductor devices like diodes, BJTs, and their applications in rectifiers, clippers, and amplifiers.
CO5: Explain the structure and operation of MOSFETs and CMOS logic circuits and their use in embedded and VLSI systems.

Module 1:

Introduction to Electrical Systems and DC Circuits
Basic concepts of electric current, voltage, power, and energy. Ohm's Law, Kirchhoff's Laws (KCL and KVL). Series and parallel circuits. Mesh and nodal analysis. Star-delta transformations. Energy sources: ideal and practical voltage and current sources.
Introduction to Power Generation and Distribution: Elementary idea of generation methods (thermal, hydro, solar, nuclear), transmission and distribution of electrical power; structure of power system; role of substations and transformers.

Module 2:

AC Circuits and Measurement
Sinusoidal voltage and current: amplitude, frequency, phase, time period. RMS and average values, form factor, peak factor. Phasor representation of sinusoidal quantities. AC analysis of pure R, L, and C components. RLC series and parallel circuits. Power in AC circuits: real, reactive, and apparent power. Power factor and its significance. Basics of three-phase systems.
Basic measuring instruments: voltmeter, ammeter, wattmeter, and energy meter.

Module 3:

Electrical Machines and Domestic Wiring
Magnetic circuits and electromagnetic induction. Transformers: principle, construction, EMF equation, efficiency, and applications. DC Machines: construction, types (motor/generator), working principle and applications. Three-phase induction motors: construction and working.
Basic concepts of domestic wiring: types of wiring, star case wiring, fuses, circuit breakers, earthing.

Module 4:

Semiconductor Devices and Applications
Semiconductors: intrinsic and extrinsic types. PN junction diode: V-I characteristics, rectifiers (half-wave and full-wave), clippers and clampers. Zener diode: breakdown and voltage regulation. BJT: construction, input-output characteristics in CE mode, application as a switch and amplifier. Basic introduction to optoelectronic devices (LED, photodiode).

Module 5:

MOSFET and CMOS Technology
MOSFET: structure, operation of nMOS and pMOS, output and transfer characteristics.

MOSFET as a switch and amplifier. CMOS logic: complementary MOS operation, CMOS inverter, static and dynamic behavior. Advantages of CMOS technology: low power, high noise margin, scalability. Applications of MOSFETs and CMOS in logic circuits, embedded systems, and VLSI design.

Labs / Practicals:

Electrical Engineering Experiments

1. Verification of Ohm's Law and Kirchhoff's Laws (KVL & KCL)
o Study linear resistive networks and validate current-voltage relationships.
2. Measurement of Power and Power Factor in a Single-Phase AC Circuit
o Calculate real, reactive, and apparent power using wattmeters and determine power factor.
3. Series and Parallel RLC Circuit Analysis
o Observe resonance and calculate impedance, current, and phase angle.
4. Load Test on Single-Phase Transformer
o Measure efficiency and voltage regulation under different loads.
5. Speed Control and Operation of a DC Motor
o Study working, direction control, and variation of speed with field/armature voltage.
6. Star Case Wiring and Testing of Light/Fan Circuit

• Perform practical wiring of a basic domestic setup using star configuration; verify connections and functionality.

Electronics Engineering Experiments

7. V-I Characteristics of PN Junction Diode
o Plot forward and reverse bias curves; identify cut-in and breakdown voltages.
8. Half-Wave and Full-Wave Rectifier with and without Filter
o Measure output DC voltage, ripple factor, and waveform using CRO.
9. Zener Diode as Voltage Regulator
o Test voltage regulation with varying load and input voltage.
10. Input and Output Characteristics of BJT in CE Configuration
o Measure current gain (β) and plot characteristics for amplification.
11. MOSFET Characteristics and CMOS Inverter Operation

References:

1. V.K. Mehta, Rohit Mehta – Principles of Electrical Engineering and Electronics, S. Chand Publishing
2. D.P. Kothari, I.J. Nagrath – Basic Electrical and Electronics Engineering, McGraw-Hill Education
3. M.S. Sukhija, T.K. Nagsarkar – Basic Electrical and Electronics Engineering, Oxford University Press

Course Code	Course Name	L	T	P	Cr
DASH250042	Engineering Physics	3	0	2	4

Course Outcomes:

CO1: Understand the phenomenon of interference in thin films and diffraction by slits
CO2: Understand the principle behind the working of LASERS and Optical Fibers
CO3: Understand concept of quantum mechanics and one-dimensional motion of particle
CO4: Understand various properties of semiconductor, superconductor and nanomaterial
CO5: Understand electric and magnetic field behavior and drive Maxwell equations

Module 1:

Wave Optics:
Concept of interference, Interference in thin films (reflected light)-Newton's Rings and Michelson's Interferometer, Application as wavelength measurement. Concept of Diffraction, Single slit Fraunhofer Diffraction, Diffraction Grating and Spectrum, Determination of Wavelength, Application of Grating as wavelength splitter.

Module 2:

Lasers & Optical Fibers:
Laser: Einstein's Theory of laser action, Einstein's coefficients, population inversion and lasing action, Properties of Laser beam, Construction and working of He-Ne and semiconductor lasers, Applications of Lasers in Science and engineering. Optical Fiber: Structure, Types, Features, Light guiding mechanism, Acceptance angle and Numerical Aperture.

Module 3:

Quantum Mechanics:
Concepts and Experiments that led to the discovery of Quantum Nature. Heisenberg uncertainty principle, Wave function and basic postulate of wave mechanics, Schrodinger time independent and time dependent wave equations, Physical interpretation of wave function and properties. The free particle problem - Particle in 1-dimensional and 3-dimensional boxes, Concept of Quantum mechanical tunneling.

Module 4:

Physics of Advanced Materials:
Types of semiconductors, Conductivity in semiconductors,
Energy Band Gap, Hall Effect: Theory and applications, Superconductors: Properties, Meissner effect, Type I & II superconductors, Applications of superconductors, Nanomaterials: Significance of nanoscale, Properties of nanomaterials, Basics of Synthesis of nanomaterials: top-down and bottom-up approach, Applications of nanomaterials, X-ray Diffraction.

Module 5:

Introduction to Electromagnetism:
Gradient, divergence and curl and their physical significance, Divergence and Curl of electrostatic and static Magnetic Fields, Faraday's law, equation of continuity, Displacement current, Maxwell's equations, Electromagnetic wave propagation in free space Flow of energy and Poynting vector.

Labs / Practicals:

1. To study the formation of Newton's rings and determine the wavelength of light (Sodium lamp/LASER).
2. To determine the wavelength of light (Sodium lamp/LASER) with the help of Michelson interferometer.
3. To determine the wavelength of prominent lines of light (mercury) by using plane transmission diffraction grating.
4. To determine specific rotation of sugar using half shade/ biquartz polarimeter.
5. To determine the dispersive power of material of a prism with the help of spectrometer.
6. To determine the height of given object with the help of sextant.

7 To determination of band gap of semiconductor using a P-N junction diode.
8. To study the Hall Effect and determination of hall coefficient and charge carrier concentration.
9. To measure the numerical aperture of an optical fiber.
10. To determine the coherence length and coherence time of laser using He -Ne laser.
11. To study the charge and discharge of a condenser and hence determine the time constant.
12. To determination of resonating frequency and bandwidth by LCR circuit.
13. To study the B-H/I-H curve and hysteresis losses in a given magnetic material.

References:

1. Halliday, Resnic and Walker, "Fundamentals of Physics", Publisher: John Wiley.
2 Beiser A, "Concepts of Modern Physics", Publisher: McGraw Hill International.
3 Ajoy Ghatak, "Optics", Publisher: Tata McGraw Hill.
4 S. O. Pillai, "Solid State Physics", Publisher: New Age Publishers.
5 A. Ghatak, K. Thyagarajan, "Introduction To Fiber Optics", Publisher: Cambridge University Press.
6 W.T Silfvast, "Laser Fundamentals", Publisher: Cambridge University Press.
7 R. Shankar, "Fundamentals of Physics", Publisher: Yale University Press, New Haven and London.
8 R. Shankar, Fundamentals of Physics II", Publisher: Yale University Press, New Haven and London.
9 David J. Griffiths, "Introduction to Electrodynamics", Publisher: Cambridge University Press.
10 K. K. Chattopadhyay and A. N. Banerjee, "Introduction to Nanoscience and Nanotechnology", Publisher: PHI Learning Pvt. Limited.
11 T. Pradeep, "NANO: The Essentials, understanding Nano science and Nanotechnology", Publisher: Tata McGraw-Hill Publishing Company Limited.

Course Code	Course Name	L	T	P	Cr
DASH250016	Communication Skills	1	0	2	2

Course Outcomes:

CO1 : Understand the need and importance of English language.
CO2 : Develop proficiency in the language.
CO3 : Able to use technology to enrich communication skills.
CO4 : Gain self-confidence with improved command over English.
CO5 : Apply English communication skills effectively in academic, professional, and social environments.

Module 1:

Communication Fundamentals
Communication ;Meaning, Process and importance of Communication, Types and Channels of communications , Scope and Significance of Listening ,Speaking, Reading, Writing Skills.

Module 2:

Writing Skills
Basics of Grammar: Parts of Speech, Uses of Tenses, Active Passive, Narration.

Module 3:

Vocabulary Building
Word Formation, Synonyms , Antonyms, Words often Confused, One Word Substitutes, Idioms and Phrasal verbs, Abbreviations of Scientific and Technical Words.

Module 4:

Speaking Skills
Introduction to Phonetic Sounds & Articulation, Word Accent, Rhythm and Intonation, Interpersonal Communication, Oral Presentation, Body Language and Voice Modulation (Para linguistics and Non Verbal), Negotiation and Persuasion, Group Discussion, Interview Techniques (Telephonic and Video conferencing).

Module 5:

Technical Writing
Job application, CV writing, Business Letters, Notices, Report Writing, Minutes, E-mail Etiquette, Blog Writing.

Labs / Practicals:

1. Introducing Oneself, Exercise on Parts of Speech & Exercise on Tense.
2. Exercise on Agreement, Narration, Active Passive Voice & Dialogue Conversation.
3. Exercise on Writing Skills and Listening Comprehension.
4. Practice of Phonemes, Word Accent, Intonation, JAM Session.
5. Individual Presentation, Extempore and Picture Interpretation.
6. Vocabulary Building Exercises (One Word Substitute, Synonyms, Antonyms, Words Often Confused etc.) & Group Discussion.

References:

1. "The Essence of Effective Communication", Ludlow R. and Panton F., Pubs: Prentice Hall.
2. “Effective Communication Skills”, Kulbhushan Kumar, Khanna Publishing House.
3. “A University Grammar of English”, Quirk R. and Sidney G., Pubs: Pearson Education.
4. “High School English Grammar”, Wren and Martin, Pubs: S. Chand & Company Ltd.
5. “Essentials of Business Communication”, Guffrey M.E., Pubs: South-Western College Publishing.
6. “Technical Communication: Principles and Practice”, Raman M. and Sharma S., Pubs: Oxford University Press.
7. “Effective Business Communication”, Rodrigues M.V., Pubs: Concept Publishing Company, Delhi.

8. “English Vocabulary in Use”, McCarthy M. and Felicity O’ Dell.

Course Code	Course Name	L	T	P	Cr
DASH250039	Engineering Mathematics II	3	1	0	4

Course Outcomes:

CO1 : Analyze the behavior of sequences and series using convergence tests and apply them in evaluating infinite processes.
CO2 : Construct Fourier series representations of periodic functions and apply them in engineering and data modeling contexts.
CO3 : Compute Laplace transforms of various functions including step and impulse functions, and use them for solving engineering problems.
CO4 : Perform inverse Laplace transforms and use them effectively to solve initial value problems in engineering models.
CO5 : Use Fourier transforms and their properties to switch between time and frequency domains, aiding AI applications such as image/signal processing and neural network optimization.

Module 1:

Sequences and Series (9 hrs)
Limit of a sequence, monotone and Cauchy sequences, properties of convergent sequences, examples. Infinite series, positive series, tests for convergence and divergence, integral test, alternating series, Leibnitz test.

Module 2:

Fourier Series (9 hrs)
Fourier Series of Periodic functions (Euler's Formulas and Dirichlet's conditions), Even and odd function expansions (Half range sine cosine series), Fourier series over arbitrary intervals (Change of interval techniques).

Module 3:

Laplace Transformation (7 hrs)
Laplace Transforms: Laplace Transform of standard functions, First shifting theorem, Second shifting theorem, Unit step function, Dirac delta function, Laplace transforms of functions when they are multiplied and divided by 't', Laplace transforms of derivatives (First and second derivative) of functions, Laplace Transformation of integral of functions, Evaluation of integrals by Laplace transforms.

Module 4:

Inverse Laplace Transformation and Applications (11 hrs)
Inverse Laplace transform by partial fractions and shifting theorem, convolution theorem (without proof). Applications: solving Initial value problems of first order by Laplace Transform method.

Module 5:

Fourier Transformation (9 hrs)
Introduction to Fourier Transform, Properties- Linearity, Time and frequency shifting property (1st and 2nd shifting theorem), Differentiation and integration in time domain and their transforms, Fourier Integral theorem (statement only), Fourier transform and its inverse, Convolution theorem (Statement and applications).

Labs / Practicals:

N/A

References:

1. Bali, N. P., & Goyal, M. A Text Book of Engineering Mathematics. Laxmi Publications (P) Ltd.
2. Kreyszig, E. Advanced Engineering Mathematics (9th ed.). John Wiley & Sons, New Delhi.
3. Ramana, B. V. Higher Engineering Mathematics. Tata McGraw-Hill, New Delhi.
4. Grewal, B. S. Higher Engineering Mathematics. Khanna Publishers, New Delhi.
5. H.K. Dass, Advanced Engineering Mathematics. S. Chand & Company Pvt. Ltd., New Delhi.
6. Jain, R. K., & Iyengar, S. R. K. Advanced Engineering Mathematics. Narosa Publishing House, New Delhi.
7. Dass, H. K., & Verma, R. Engineering Mathematics. S. Chand & Company Pvt. Ltd., New Delhi.
8. Pundir, R. S., & Pundir, S. Engineering Mathematics. Laxmi Publications (P) Ltd.
9. Ghosh, S. C. Engineering Mathematics. PHI Learning Pvt. Ltd.
10. Gupta, C. S., & Malik, A. Mathematical Analysis and Applications. Wiley Eastern / Narosa Publishing House

Course Code	Course Name	L	T	P	Cr
DOEE250059	Digital Electronics	3	0	2	4

Course Outcomes:

CO1: Perform conversions between number systems and design logic functions using basic and universal gates.
CO2: Simplify Boolean expressions using Karnaugh Maps and Boolean algebra for circuit minimization.
CO3: Design and analyze combinational logic circuits like adders, subtractors, multiplexers, and decoders.
CO4: Understand and implement sequential circuits such as flip-flops, counters, and shift registers.
CO5: Understand the architecture and applications of memory units and programmable logic devices.

Module 1:

Number System and Logic Gates: Representation and interconversion among binary, octal, decimal, and hexadecimal number systems, Binary arithmetic operations: addition, subtraction, multiplication, and division, Binary Coded Decimal (BCD), Excess-3, and Gray codes, Signed and unsigned binary number representation,1's and 2's complement techniques for binary subtraction, Code conversions: Binary to BCD, Excess-3, and Gray codes, and vice versa, Logic gates: AND, OR, NOT, NAND, NOR, XOR, XNOR, Universal gates and implementation of basic logic functions using universal gates.

Module 2:

Boolean Algebra and Logic Simplification: Boolean laws and theorems, DeMorgan’s theorems, SOP and POS forms, Canonical forms, Karnaugh Map (2, 3, 4 variables) – minterms and maxterms, Quine-McCluskey method (basic idea), Logic expression simplification and gate minimization.

Module 3:

Combinational Logic Circuits: Adders: Half adder, full adder, ripple carry adder, carry look-ahead adder, Subtractors: Half and full subtractors, Comparators, Multiplexers (MUX) and De-multiplexers (DEMUX) Encoders and Decoders, BCD to 7-segment decoder, Design of combinational logic using MUX, DEMUX.

Module 4:

Sequential Logic Circuits: Latches and Flip-flops: SR, D, T, JK, Master-Slave, Characteristic tables and excitation tables, Edge and level triggering, Timing diagrams, Counters: Asynchronous (ripple) and Synchronous counters, Up/down counters, modulo-N counters, Shift registers: SISO, SIPO, PISO, PIPO, Ring counter and Johnson counter, Mealy and Moore Finite State Machines (FSMs).

Module 5:

Memory & Logic Families: Classification of memories: RAM, ROM, PROM, EPROM, EEPROM, Memory decoding and address space, Overview of programmable logic devices: PLA, PAL, CPLD, FPGA (introductory), Logic families: TTL, CMOS. Characteristics: propagation delay, power dissipation, noise margin, fan-out, fan-in.

Labs / Practicals:

1. Verification of Logic Gates: Implement and verify the truth tables of AND, OR, NOT, NAND, NOR, XOR, and XNOR gates.
2. Universal Gates Implementation: Realize basic logic functions using only NAND or NOR gates.
3. Boolean Expression Simplification: Simplify given Boolean expressions and implement the minimized circuits using logic gates.
4. Design of Half Adder and Full Adder: Construct and test circuits for half and full adders using logic gates.
5. Design of Half Subtractor and Full Subtractor: Implement and verify half and full subtractor circuits.
6. Multiplexer and Demultiplexer: Design and test 4:1 MUX and 1:4 DEMUX circuits.
7. Flip-Flop Implementations: Implement and observe SR, D, T, and JK flip-flops using basic gates or ICs.
8. Counters: Design and test asynchronous and synchronous up/down counters (e.g., mod-8, mod-10).
9. Shift Registers: Construct and analyze SIPO, SISO, PIPO, and PISO shift registers.

10. BCD to 7-Segment Display: Design a decoder to display BCD inputs on a 7-segment LED display.

References:

1. M. Morris Mano, Michael D. Ciletti – Digital Design, Pearson Education
2. R.P. Jain – Modern Digital Electronics, McGraw-Hill Education
3. Thomas L. Floyd – Digital Fundamentals, Pearson Education
4. John F. Wakerly – Digital Design: Principles and Practices, Pearson
5. Tokheim – Digital Electronics: Principles and Applications, McGraw-Hill Education

Course Code	Course Name	L	T	P	Cr
DCSA250016	Computer Aided Graphics Design	1	1	4	4

Course Outcomes:

CO1: Understand the fundamental principles of computer graphics and graphic design.
CO2: Apply transformations, projections, and rendering techniques for 2D and 3D modeling.
CO3: Develop proficiency in using CAD and vector graphic design tools for technical drawings and creative graphics.
CO4: Demonstrate the ability to perform digital editing, modeling, and visualization using software like AutoCAD, Adobe Illustrator, or Blender.
CO5: Create complete design projects from concept to digital representation, including rendering and documentation.

Module 1:

Basics of Computer Graphics: Introduction to Computer Graphics, Applications of Graphics in Design, Pixel and Resolution Concepts, Raster vs Vector Graphics, Coordinate Systems, Line and Circle Drawing Algorithms (DDA, Bresenham).

Module 2:

2D Transformations and Viewing: Basic 2D Transformations: Translation, Rotation, Scaling, Reflection, Shearing. Homogeneous Coordinates, Matrix Representation of Transformations, Composite Transformations. Windowing and Clipping Techniques: Cohen–Sutherland, Liang–Barsky.

Module 3:

3D Graphics and Modeling: 3D Coordinate Systems, 3D Transformations, Projection Techniques (Perspective and Parallel), Hidden Surface Removal, Basics of 3D Modeling and Visualization, Introduction to 3D Modeling Tools (e.g., Blender/AutoCAD).

Module 4:

Graphic Design Tools and Techniques: Introduction to Design Software: Adobe Illustrator/CorelDRAW/Inkscape. Vector Graphics Tools: Pen, Shape, Path Editing. Typography, Colors, Layers, and Effects. Designing Logos, Brochures, and Posters.

Module 5:

CAD and Real-World Applications: Introduction to AutoCAD or Equivalent CAD Tool, Drawing 2D and 3D Objects, Layers and Blocks, Dimensioning and Annotation, Rendering and Exporting Drawings. Application of CAD in Engineering Drawings, Architecture, and Product Design.

Labs / Practicals:

Module 1 & 2 – 2D Graphics and Transformations
1. Implement DDA Line Drawing Algorithm using C/C++ or Python.
2. Perform 2D Transformations (Translation, Rotation, Scaling, Reflection) on a geometric shape.
3. Implement Cohen–Sutherland Line Clipping Algorithm for windowing and clipping.

Module 3 – 3D Graphics and Modeling
4. Create and Transform 3D Objects (e.g., Cube, Sphere) using Blender.
5. Apply Perspective and Parallel Projections to a 3D model and visualize the differences.

Module 4 – Graphic Design Tools
6. Design a Logo using vector graphic tools in Adobe Illustrator or Inkscape.
7. Create a Poster or Brochure using typography, color themes, and shapes.

Module 5 – CAD Applications
8. Draw a 2D Engineering Component using AutoCAD with layers and dimensions.
9. Model a Simple 3D Object in AutoCAD and apply rendering.

10. Export and Annotate Drawings from AutoCAD for documentation or print-ready formats.

References:

1. Donald Hearn and M. Pauline Baker – Computer Graphics with OpenGL, Pearson Education.
2. D. F. Rogers and J. A. Adams – Mathematical Elements for Computer Graphics, McGraw-Hill.
3. Sham Tickoo – Learning AutoCAD: A Project-Based Tutorial, CADCIM Technologies.
4. Adobe Creative Team – Adobe Illustrator Classroom in a Book, Adobe Press.
5. Blender Foundation – Blender Basics: Classroom Tutorial Book, Blender Foundation.

Course Code	Course Name	L	T	P	Cr
DASH250030	Economics for Engineers	1	1	0	2

Course Outcomes:

CO1 : Understand basic economic concepts, including demand, supply, cost, and market structures.
CO2 : Analyze the economic feasibility of engineering projects using cost-benefit and break-even analysis.
CO3 : Evaluate different investment alternatives using engineering economic tools like NPV, IRR, and Payback Period.
CO4 : Apply knowledge of macroeconomic indicators and fiscal/monetary policies in the context of engineering.
CO5 : Develop the ability to assess the economic impact of engineering decisions in real-world scenarios.

Module 1:

Basic Economic Concepts
Introduction to Economics: Micro vs Macro, Utility, Demand and Supply Analysis, Elasticity of Demand and Supply, Equilibrium, Law of Diminishing Returns, Opportunity Cost.

Module 2:

Cost and Production Analysis
Types of Costs: Fixed, Variable, Marginal, Average, Total, Cost-Output Relationship, Break-even Analysis, Law of Variable Proportions, Returns to Scale, Cost-Volume-Profit Analysis.

Module 3:

Market Structures and Pricing
Types of Market: Perfect Competition, Monopoly, Oligopoly, Monopolistic Competition, Pricing Strategies, Price Discrimination, Government Policies and Pricing.

Module 4:

Engineering Economics
Time Value of Money, Present Worth and Future Worth, Net Present Value (NPV), Internal Rate of Return (IRR), Payback Period, Comparison of Investment Alternatives, Depreciation and Inflation Considerations.

Module 5:

Macroeconomic Concepts
National Income, GDP, GNP, Inflation, Deflation, Unemployment, Fiscal and Monetary Policy, Role of RBI and Government in Economic Planning, Economic Indicators Relevant to Engineers.

Labs / Practicals:

N/A

References:

1. R. Panneerselvam, Engineering Economics, PHI Learning.
2. D.M. Mithani, Managerial Economics, Himalaya Publishing House.
3. M.L. Jhingan, Microeconomic Theory, Vrinda Publications, Latest Edition.
4. Chan S. Park, Contemporary Engineering Economics, Pearson Education.
5. Paul A. Samuelson and William D. Nordhaus, Economics, McGraw Hill.

Course Code	Course Name	L	T	P	Cr
DASH250028	Discrete Mathematics	3	1	0	4

Course Outcomes:

CO1 : Apply propositional and predicate logic in the formalization of mathematical and computer science problems.
CO2 : Solve number-theoretic and combinatorial problems using modular arithmetic and binomial identities.
CO3 : Implement counting strategies and solve discrete problems using pigeonhole and inclusion-exclusion principles.
CO4 : Analyze and model partially ordered sets and apply lattice and Boolean algebra in digital logic design.
CO5 : Understand and apply graph theory concepts to real-world problems such as routing and the Travelling Salesman Problem.

Module 1:

Logic and Set Theory (8 hrs)
Propositional and Predicate Logic: Statements, Predicates and quantifiers, nested quantifiers, Logical connectives, truth tables, logical equivalences
Set Theory: Sets and set operations, Venn diagrams, set identities
Functions: definitions, domain, codomain, range, Inverse and composition of functions.

Module 2:

Number Theory and Arithmetic (8 hrs)
Basic Number Theory: Division algorithm, divisibility, Prime numbers and prime factorization, Greatest Common Divisor (GCD) and Least Common Multiple (LCM)
Modular Arithmetic: Congruence relations and properties, Applications in computing and cryptography.

Module 3:

Combinatorics and Binomial Principles (6 hrs)
Permutations and Combinations, Pigeonhole Principle: Basic and generalized form, Applications in problem-solving, Inclusion-Exclusion Principle, Binomial Theorem (without proof): General term, middle term, Applications in algebra and combinatorics, Pascal's Identity: Construction and properties.

Module 4:

Lattices and Boolean Algebra (10 hrs)
Partially ordered sets (Posets), Hasse diagrams
Lattices: Definition and basic properties, Bounded, distributive, and complemented lattices
Boolean algebra: Definition, identities, and operations
Boolean functions: Sum of Products (SOP), Product of Sum (POS) forms and simplification basics.

Module 5:

Relations and Graph Theory (8 hrs)
Relations: Types and properties: reflexivity, symmetry, transitivity, Representations of relations, Equivalence relations
Graphs: Basic terminology and types of graphs, Graph representations: adjacency matrix, incidence matrix, adjacency list, Connectivity, Eulerian and Hamiltonian paths and circuits, Travelling Salesman Problem (TSP): introduction, formulation.

Labs / Practicals:

N/A

References:

1. Rosen Kenneth H., Discrete Mathematics and its Applications, McGraw Hill.
2. Balakrishnan S., Discrete Mathematics, Laxmi Publications.
3. Babu Ram, Discrete Mathematics, Pearson Education / Vikas Publishing.
4. Kolman Bernard, Busby Robert C. and Ross Sharon Cutler, Discrete Mathematical Structures, Pearson Education.
5. Tremblay J.P. and Manohar R., Discrete Mathematics, McGraw Hill.
6. Deo Narsingh, Graph Theory with Applications, PHI Learning.
7. Liu C.L., Elements of Discrete Mathematics, McGraw Hill.

Course Code	Course Name	L	T	P	Cr
DCSE250055	Data Structures	3	0	2	4

Course Outcomes:

CO1: Remember the different types of linear and non-linear data structures.

CO2: Understand how structuring of data can reduce the average access time.

CO3: Apply concrete data structures to implement abstract data structures.

CO4: Analyze the fundamental operations supported by various data structures.

CO5: Evaluate the number of steps required to do an operation on a data structure.

CO6: Create efficient implementations that are suitable for real-world applications.

Module 1:

Multi-dimensional arrays, addressing in row-major and column-major orderings.
Searching: linear search, binary search, uniform binary search, interpolation search.
Sorting by comparison: selection, bubble, insertion, Shell sort, quicksort, mergesort.
Non-comparison sorting: counting sort, pigeonhole sort, radix sort (LSD and MSD).

Module 2:

Linked lists types: linear and circular, unidirectional and bidirectional.
Operations on linked lists: create, insert, delete, traverse, reverse, find.
Open address hashing, collision, linear probing, quadratic probing, double hashing.
Hashing with separate chaining, load factor, applications of hash tables.

Module 3:

Abstract data structures, concrete implementation using arrays and linked lists.
Stacks: LIFO access, operations (push, pop, top), call stack, expression evaluation.
Queues: FIFO access, operations (enqueue, dequeue, peek), linear queue, ring buffer.
Double-ended queue (input-restricted, output-restricted), priority queue, DEPQ.

Module 4:

Graphs (undirected and directed), adjacency matrix, adjacency list, weighted graphs.
Graph traversals: breadth-first search, depth-first search, iterative deepening search.
Graph connectivity, directed acyclic graphs, topological sorting, spanning trees.
Arborescence, branching factor, binary trees (skew, strict, complete, perfect).

Module 5:

Tree traversal: pre-order (NLR), in-order (LNR), post-order (LRN), reverse variants.
Binary heap (min-heap and max-heap), Fibonacci heap, binomial heap, heapsort.
Binary search tree (BST), sorting with BST, threaded BST (single and double).
Disjoint-set data structure, operations: makeset, find, union (by rank or by size).

Labs / Practicals:

Each data structure and supporting algorithms should be implemented in C or Python.

01. Use arrays to store univariate polynomials for doing addition and multiplication.

02. Write a function to find min/max element and use it to implement selection sort.

03. Implement an adaptive variant of bubble sort that runs in linear time for best case.

04. Implement insertion sort with a type-agnostic design (like qsort from stdlib.h).

05. Perform non-comparative sorting with counting sort and radix sort (LSD and MSD).

06. Create unidirectional and bidirectional linked lists with linear and circular variants.

07. Create open address hash tables with linear probe, quadratic probe, double hashing.

08. Create hash table with an array of pointers for separate chaining using linked lists.

09. Implement stack, queue (linear and circular), deque using arrays and linked lists.

10. Convert among prefix, infix, postfix expressions, and evaluate the postfix notation.

11. Perform breadth-first traversal, depth-first traversal, topological sort on a digraph.

12. Implement priority queue using binary max heap, supporting insertion and deletion.

13. Heapify an array of strings in linear time and do heap sort in lexicographic order.

14. Perform insert and delete on a binary search tree threaded for in-order traversal.

15. Implement disjoint-set data structure supporting makeset, find, union operations.

References:

1. "Think Data Structures" by Allen Benjamin Downey
https://greenteapress.com/thinkdast/thinkdast.pdf

2. "Open Data Structures: An Introduction" by Pat Morin
https://www.aupress.ca/app/uploads/120226_99Z_Morin_2013-Open_Data_Structures.pdf

3. "An Open Guide to Data Structures and Algorithms" by Paul W. Bible and Lucas Moser
https://pressbooks.palni.org/anopenguidetodatastructuresandalgorithms

4. "Computer Science II" by Chris Bourke
https://cse.unl.edu/~cbourke/ComputerScienceTwo.pdf

5. "Data Structures and Algorithms" by Alfred Vaino Aho, John Edward Hopcroft, Jeffrey David Ullman
https://dl.acm.org/doi/10.5555/577958

6. "Data Structures and Algorithm Analysis in C (Second Edition)" by Mark Allen Weiss
https://dl.acm.org/doi/10.5555/248735

Course Code	Course Name	L	T	P	Cr
DOEE250044	Computer Organization and Architecture	3	0	2	4

Course Outcomes:

CO1: Explain processor fundamentals, instruction representation, and hardware operations including arithmetic and logical functions.
CO2: Analyze instruction set architectures, addressing modes, and computer arithmetic with parallelism and SIMD extensions.
CO3: Design basic datapath and pipelined architectures, and evaluate instruction-level parallelism and hazards in modern processors.
CO4: Demonstrate memory mapping techniques, cache design, memory hierarchy, and virtual memory mechanisms.
CO5: Evaluate memory management strategies including RAID levels, disk storage, multithreading, and multiprocessor systems.

Module 1:

Introduction of Processor:Introduction, Technologies for building Processors and Memory, Performance, The Power Wall, Operations of the Computer Hardware, Operands Signed and Unsigned numbers, Representing Instructions, Logical Operations, Instructions for Making Decisions

Module 2:

Instructions Set:MIPS Addressing for 32-Bit Immediate and Addresses, Parallelism and Instructions: Synchronization, Translating and Starting a Program, Addition and Subtraction, Multiplication, Division, Floating Point, Parallelism and Computer Arithmetic: Sub word Parallelism, Streaming SIMD Extensions and Advanced Vector Extensions in x86.

Module 3:

Architecture Building Block: Logic Design Conventions, building a Datapath, A Simple Implementation Scheme, overview of Pipelining, Pipelined Datapath, Data Hazards: Forwarding versus Stalling, Control Hazards, Exceptions, Parallelism via Instructions, The ARM Cortex – A8 and Intel Core i7 Pipelines, Instruction –Level Parallelism and Matrix Multiply Hardware Design language.

Module 4:

Memory Mapping: Memory Technologies, Basics of Caches, Measuring and Improving Cache Performance, dependable memory hierarchy, Virtual Machines, Virtual Memory, Using FSM to Control a Simple Cache, Parallelism and Memory Hierarchy: Redundant Arrays of Inexpensive Disks, Advanced Material: Implementing Cache Controllers.

Module 5:

Memory Management: Disk Storage and Dependability, RAID levels, performance of storage systems, Introduction to multi- threading clusters, message passing multiprocessors.

Labs / Practicals:

Module 1 & 2: Processor Basics and Instruction Set
1. Simulation of Basic Arithmetic Operations in MIPS Assembly
o Objective: Write MIPS assembly code to perform addition, subtraction, multiplication, and division of two integers.
o Tool: MIPS Simulator (e.g., QtSpim, MARS)
2. Binary and Hexadecimal Conversion & Representation
o Objective: Implement a program (in C or Assembly) to convert between binary, hexadecimal, and decimal formats; demonstrate signed and unsigned number representations.

3. Instruction Encoding and Decoding
o Objective: Manually encode and decode sample MIPS instructions and verify using a simulator.

Module 3: Datapath and Pipelining
4. Design a Basic Arithmetic Logic Unit (ALU) Using VHDL/Verilog
o Objective: Implement an ALU capable of performing AND, OR, ADD, SUB, and SLT operations.
o Tool: Xilinx Vivado / ModelSim / Logisim
5. Simulate Single-Cycle and Multi-Cycle Datapaths
o Objective: Model a simplified processor datapath (single-cycle and pipelined) and compare performance.
o Tool: Digital Simulator / Logisim / Ripes (RISC-V simulator)
6. Pipeline Hazard Detection and Resolution
o Objective: Simulate and analyze data and control hazards in a pipelined processor; demonstrate stalling and forwarding.

Module 4 & 5: Memory Mapping and Management
7. Cache Memory Simulation
o Objective: Simulate direct-mapped, fully-associative, and set-associative caches; compute hit/miss ratio.
o Tool: Custom Python/C program or existing cache simulator tools.
8. Implementation of Virtual Memory using Paging
o Objective: Simulate address translation using page tables; demonstrate page faults and TLBs.
o Tool: Python or Java-based simulation
9. Disk Scheduling Algorithm Simulation
o Objective: Implement and compare performance of FCFS, SSTF, SCAN, and LOOK disk scheduling algorithms.
10. RAID Level Simulation and Performance Comparison
• Objective: Simulate different RAID levels (RAID 0, 1, 5) and evaluate redundancy, speed, and fault tolerance.

References:

1. David A. Patterson and John L. Hennessey, “Computer organization and design, The Hardware/Software interface”, Morgan Kauffman / Elsevier, Fifth edition, 2014
2. V. Carl Hamacher, Zvonko G. Varanesic, and Safat G. Zaky, “Computer Organization “, 6 th edition, McGraw-Hill Inc, 2012.
3. William Stallings, “Computer Organization and Architecture”, 8th Edition, Pearson Education, 2010

Course Code	Course Name	L	T	P	Cr
DOEE250022	Assembly Language and Microprocessors	3	0	2	4

Course Outcomes:

CO1. Describe the architecture, pin configuration, and functional units of 8085/8086 microprocessors.
CO2. Explain various addressing modes and instruction sets of 8085/8086 microprocessors.
CO3. Write, assemble, and execute simple assembly language programs for arithmetic, logic, and data transfer operations.
CO4. Design small embedded system applications using microprocessor/microcontroller programming.

Module 1:

Introduction to Assembly Language:
Basics of low-level programming, Machine language vs assembly language, Role of assembler, Structure of an assembly language program, Instruction formats and addressing modes

Module 2:

Introduction to Microprocessor: Definition of Microprocessor, Bit, Byte, Nibble, Instruction, Mnemonics, Assembly Level Programming, Assembler, Hexadecimal to Decimal conversion and vice-versa, Evolution of microprocessors, Block diagram of microprocessor based digital computer: functions of each Block, Application of microprocessors.

Module 3:

Architecture and Organization of 8085: Central Processing Unit, Accumulator, General Purpose Register, Status Register, ALU, Program Counter (PC), Stack Pointer (SP), Control Unit, The Clock, Reset, Interrupt, Timing and Control Unit, Pin Configuration of 8085: functions of each pin.

Module 4:

8085 Instruction set: Machine language instruction format : Single byte, two byte, three byte instructions, Various addressing modes, Data transfer operation and instruction, Arithmetic operation and instruction, Logical operation and instruction, Branch operation and instruction, Stack operation and instruction, Input/Output and machine control operation and instruction

Module 5:

Introduction to 16 bit microprocessors and other advanced microprocessors, 8086 pin diagram: Functions of each pin, 8086 Architecture, various registers: General Purpose, Segment, Index,
Base, Pointers, flags; Interrupts, 8086 Addressing Modes, Instruction Format.

Labs / Practicals:

1. To write a program to add two 8 bit number using 8085 microprocessor.
2. To write a program to subtract two 8 bit numbers using 8085 microprocessor.
3. To write an assembly language program for multiplication of two 8-bit numbers using 8085 instructions.
4. To write an assembly language program for division of two 8-bit numbers using 8085 instructions.
5. To write an assembly language program for addition of two 16-bit number without carry using 8086 instruction.
6. To write an assembly language program for 1’s complement of an 8-bit numbers using 8085 instructions.
7. To write an assembly language program for 2’s complement of an 8-bit numbers using 8085 instructions.
8. To write an assembly language program for multiplication of two 16-bit number using 8086 instructions.

References:

1. Microprocessor and Microcomputer: Gaonkar (PRI)
2. Microprocessor and microcontroller: Senthilkumar (Oxford)

3. 8085 Microprocessor, programming and interface: Srinath (PHI)
4. Microprocessors: Theory and application: MahammadRafiquzzaman(PHI)

Course Code	Course Name	L	T	P	Cr
DASH250104	Technical Presentation and Report Writing	0	0	4	2

Course Outcomes:

CO1 : Understand the principles and formats of technical communication including different types of reports and presentations.
CO2 : Prepare clear, concise, and well-structured technical documents for academic and professional purposes.
CO3 : Design and deliver effective oral presentations using appropriate tools and visual aids.
CO4 : Demonstrate proficiency in grammar, technical vocabulary, and formal style required in technical writing.
CO5 : Collaborate in teams to produce technical content and receive peer feedback constructively.

Module 1:

Introduction to Technical Communication
Principles and characteristics of technical communication, Purpose and types of technical documents, Features of a good technical report, Audience analysis.

Module 2:

Writing Skills for Technical Documents
Structure of reports: Title Page, Abstract, Table of contents, Introduction, Body, Conclusion, and References. Writing instructions, lab reports, Project Reports, and Proposals. Use of charts, graphs, and tables.

Module 3:

Grammar and Language in Technical Writing
Use of formal tone and objective language, sentence construction, common errors, punctuation, and vocabulary building for technical writing. Editing and proofreading techniques.

Module 4:

Preparing Technical Presentations
Planning a presentation, structuring content, use of slides, visual aids, and multimedia tools (e.g., PowerPoint, Prezi). Non-verbal communication and handling audience questions.

Module 5:

Practice and Evaluation
Student individual and group presentations on technical topics, peer reviews, instructor feedback. Writing mini technical reports based on lab/project work or case studies.

Labs / Practicals:

1. Analyzing Samples of Technical Documents
Objective: Identify the structure and features of different technical documents (e.g., lab report, project report, instruction manual).
2. Writing a Lab Report Based on Practical Work
Objective: Write a complete lab report including abstract, objectives, procedure, observations, and conclusion.
3. Creating a Project Proposal Document
Objective: Prepare a short technical proposal on a selected topic including problem statement, objectives, methodology, and timeline.
4. Designing Visual Elements (Graphs, Charts, Tables)
Objective: Use data to create visual aids using Excel/PowerPoint and integrate them into a technical report.
5. Grammar and Proofreading Workshop
Objective: Identify and correct errors in sample texts focusing on sentence construction, punctuation, and vocabulary.
6. Preparing and Delivering an Individual Presentation
Objective: Create and present a 5-minute talk on a technical topic using PowerPoint or Prezi.

7. Group Presentation on a Technical Case Study
Objective: Collaboratively research and present findings on a real-world technical issue or innovation.
8. Peer Review and Feedback Session
Objective: Evaluate and provide constructive feedback on peers' written reports and oral presentations.
9. Editing and Improving a Poorly Written Report
Objective: Revise and rewrite a given flawed report to meet academic/professional standards.
10. Writing a Mini Technical Report on a Chosen Topic
Objective: Submit a short report (3–5 pages) on a lab experiment, research article, or emerging technology.

References:

1. Meenakshi Raman and Sangeeta Sharma – Technical Communication: Principles and Practice, Oxford University Press.
2. M. Ashraf Rizvi – Effective Technical Communication, Tata McGraw-Hill.
3. Sharma, R.C. and Krishna Mohan – Business Correspondence and Report Writing, Tata McGraw-Hill.
4. Lesikar, Raymond V. et al. – Basic Business Communication, Tata McGraw-Hill.
5. David F. Beer and David McMurrey – A Guide to Writing as an Engineer, Wiley India.

Course Code	Course Name	L	T	P	Cr
DCSE250120	Theory of Computation	3	1	0	4

Course Outcomes:

CO1. Explain the basic concepts of automata, formal languages, and grammar types.
CO2. Design finite automata (deterministic and non-deterministic) and regular expressions for recognizing regular languages.
CO3. Design regular expressions for describing simple patterns and languages.
CO4. Understand and design pushdown automata for basic context-free languages.

Module 1:

Mathematical Preliminaries: Sets, Relations and Functions (Brief Discussion), Graphs, Trees. ,Strings and their properties: Definition, operation on strings, palindrome, prefix & suffix of a string, Levi theorem (Statement only), Terminal & Non-terminal symbol

Module 2:

Definition of an Automaton: Definition of finite Automaton, Block diagram of finite Automaton, Transition system, Properties of Transition Functions, Acceptability of a string by Finite Automaton. Definition of DFA and NDFA, The equivalence of DFA and NDFA. Mealy and Moore machine.

Module 3:

Formal Language: Concept of a language, Definition of a grammar, Language generated by a grammar (definition with application). Chomsky classification of languages (definition), Relation between the classified languages. Recursive and recursively enumerable set (definitions).

Module 4:

Regular Sets & Regular Grammar: Definition of Regular expression and regular set, Identities of regular expressions
Relation between regular expression and finite automata, Transition system containing /\-moves (application), Conversion of Non-deterministic systems to deterministic system (application), Construction of finite automata equivalent to a regular expression (with application)

Module 5:

Context-Free Languages & Pushdown Automata: Introduction – Definition – Derivation trees (Definitions & application) – Ambiguity in
CFG, Basic definition of PDA

Labs / Practicals:

NA

References:

1. "Foundations of Computation (Second Edition)" by Carol Critchlow and David Eck
https://math.hws.edu/FoundationsOfComputation/FoundationsOfComputation_2.3.2_8.5x11.pdf

2. "Automata Theory: An Algorithmic Approach" by Francisco Javier Esparza Estaun and Michael Blondin
https://michaelblondin.com/automata/files/book_authors.pdf

3. "Automata and Computability" by Dexter Campbell Kozen
https://www.cs.cornell.edu/kozen/Papers/481.pdf

4. "Introduction to Automata Theory, Languages, And Computation (First Edition)" by John Edward Hopcroft and Jeffrey David Ullman
http://infolab.stanford.edu/~ullman/ialc.html

5. "Introduction to the Theory of Computation (Third Edition)" by Michael Fredric Sipser
https://dl.acm.org/doi/10.5555/524279

6. "An Introduction to Formal Languages and Automata (Fifth Edition)" by Peter Linz
https://dl.acm.org/doi/10.5555/1995326

Course Code	Course Name	L	T	P	Cr
DCSE250010	Design and Analysis of Algorithms	3	0	2	4

Course Outcomes:

CO1: Understand the fundamental concepts of algorithm design and analysis.
CO2: Implement and analyze algorithms using different design paradigms.
CO3: Have the mathematical foundation in analysis of algorithms
CO4: Understand different algorithmic design strategies like Divide and conquer, Dynamic programming, greedy Algorithms, backtracking.
CO5: Analyze the efficiency of algorithms using time and space complexity theory

Module 1:

Notion of Algorithm, Role of algorithms in computing, Growth of functions- Asymptotic notations, Mathematical Analysis of Non-Recursive and Recursive Algorithms, Brute Force Approaches - Introduction, Selection Sort and Bubble Sort, Sequential Search. Brute Force String Matching.

Module 2:

Divide and Conquer: General Method, Defective Chess Board, Binary Search, Merge Sort, Quick Sort, Heap sort and its performance.

Module 3:

The Greedy Method: The General Method, Amortized Complexity, Knapsack Problem, Job Sequencing with Deadlines, Minimum-Cost Spanning Trees: Prim's Algorithm, Kruskal's Algorithm; Single Source Shortest Paths;
Dynamic Programming:
The General Method, Matrix Chain Multiplication, Longest Common Subsequence, Warshall's Algorithm, Floyd's Algorithm for the All-Pairs Shortest Paths Problem, Single-Source Shortest Paths: General Weights, 0/1 Knapsack, The Traveling Salesperson problem.

Module 4:

Decrease-And-Conquer Approaches, Space-Time Tradeoffs
Decrease and Conquer Approaches: Introduction, Insertion Sort, Depth First Search and Breadth First Search, Topological Sorting, Space-Time Tradeoffs: Introduction, Sorting by Counting, Input Enhancement in String Matching.

Module 5:

Limitations of Algorithmic Power and Coping with them: Lower Bound Arguments, Decision Trees, P, NP, and NP-Complete Problems, Challenges of
Numerical Algorithms;
Coping with Limitations of Algorithmic Power:
Backtracking: n-Queens problem, Hamiltonian Circuit Problem, Subset – Sum Problem. Branch-and-Bound: Assignment Problem, Knapsack Problem, Traveling Salesperson Problem. Approximation Algorithms for NP-Hard Problems, Traveling Salesperson Problem, Knapsack Problem.

Labs / Practicals:

Each algorithm should be implemented in C or Python, and operate on runtime inputs.

01. Implement a linear time greedy algorithm for the fractional knapsack problem.

02. Implement a greedy algorithm for optimal solution of job scheduling with deadlines.

03. Implement Kruskal's algorithm for minimum spanning tree of a connected graph.

04. Implement Prim’s algorithm for minimum spanning tree of a connected graph.

05. Implement Dijkstra’s algorithm for single-source shortest paths (no negative edge).

06. Implement Bellman-Ford algorithm for single-source shortest paths on a digraph.

07. Implement the Euclidean algorithm to find the GCD of two non-negative integers.

08. Implement the binary exponentation algorithm for efficient modular exponenation.

09. Implement binary search, uniform binary search, and interpolation search.

10. Implement quicksort with Hoare partitioning and Lomuto partioning schemes.

11. Implement an adaptive variant of mergesort that runs in linear time for best case.

12. Implement Floyd-Warshall algorithm for all-pairs shortest-path on a weighted graph.

13. Use dynamic programming to find optimal grouping in matrix chain multiplication.

14. Implement a solver for n-queens puzzle, along with a polynomial-time verifier.

15. Implement a polynomial time solver for 2-SAT problem expressed as Krom formula.

References:

1. "Computer Science III" by Chris Bourke
https://cse.unl.edu/~cbourke/ComputerScienceThree.pdf

2. "Algorithms in C (Third Edition)" by Robert Sedgewick
https://www.oreilly.com/library/view/algorithms-in-c/9780768685312

3. "Algorithms Illuminated (Omnibus Edition)" by Timothy Avelin Roughgarden
https://www.algorithmsilluminated.org

4. "The Algorithm Design Manual (Third Edition)" by Steven Sol Skiena
https://www.algorist.com/Algorist_ed2

5. "The Design and Analysis of Computer Algorithms" by Alfred Vaino Aho, John Edward Hopcroft, Jeffrey David Ullman
https://dl.acm.org/doi/10.5555/578775

6. "Computers and Intractability; A Guide to the Theory of NP-Completeness" by Michael Randolph Garey and David Stifler Johnson
https://dl.acm.org/doi/10.5555/578533

Course Code	Course Name	L	T	P	Cr
DCSE250102	Operating Systems	3	0	2	4

Course Outcomes:

CO1: Understand the basics of operating systems like kernel, shell, types and services of operating systems.
CO2:Understand the concept of program, process and thread and analyse various CPU scheduling Algorithms.
CO3: Describe and analyse the memory management and its allocation policies.
CO4: Understand the issues related to file system interface and implementation.
CO5: Understand disk management and explain disk scheduling algorithms for better utilization of external memory.
6: Configure OS in an efficient and secure manner.

Module 1:

Introduction: Overview of Operating System, basic concepts, UNIX/LINUX Architecture, Kernel, services and systems calls, system programs.

Module 2:

Process Management and Memory management: Process Management: Process concepts, operations on processes, IPC, Process Scheduling, Multithreaded programming. Memory management: Memory allocation, Swapping, Paging, Segmentation, Virtual Memory, various faults.

Module 3:

File management: Concept of a file, access methods, directory structure, file system mounting, file sharing and protection, file system structure and implementation, directory implementation, free space management, efficiency and performance. Different types of file systems.

Module 4:

I/O System: Mass storage structure - overview, disk structure, disk attachment, disk scheduling algorithms, swap space management, RAID types.

Module 5:

OS Security: Authentication, Access Control, Access Rights, System Logs

Labs / Practicals:

1. Revision practice of various commands like man, cp, mv, ln, rm, unlink, mkdir, rmdir, etc and many more that were learnt in IT Workshop course and later.
2. Implement two way process communication using pipes.
3. Implement message queue form of IPC
4. Implement shared memory and semaphore form of IPC
5. Simulate the CPU scheduling algorithms - Round Robin, SJF, FCFS, priority
6. Simulate all FIFO Page Replacement Algorithm using C program
7. Simulate all LRU Page Replacement Algorithms using C program
8. Simulate Paging Technique of Memory Management
9. Practice various commands/utilitiessuch as catnl, uniq, tee, pg, comm, cmp, diff, tr, tar, cpio, mount, umount, find, umask, ulimit, sort, grep, egrep,fgrep cut, paste, join, du, df , ps, who, etc and many more.

References:

1. "Operating System Concepts (Tenth Edition)" by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
https://os-book.com/OS10/index.html

2. "Operating Systems: Internals and Design Principles (Ninth Edition)" by William Stallings
http://williamstallings.com/OperatingSystems

3. "Modern Operating Systems (Fifth Edition)" by Andrew Stuart Tanenbaum and Herbert Bos
https://dl.acm.org/doi/10.5555/2655363

4. "Operating System Design: The Xinu Approach (Second Edition)" by Douglas Earl Comer
https://www.oreilly.com/library/view/operating-system-design/9781498712439

5. "The Design of the UNIX Operating System" by Maurice J. Bach
https://dl.acm.org/doi/10.5555/8570

6. "Advanced Programming in the UNIX Environment (Third Edition)" by William Richard Stevens and Stephen A. Rago
http://www.apuebook.com/apue3e.html

7. "Beej's Guide to Interprocess Communication" by Brian Hall
https://beej.us/guide/bgipc/pdf/bgipc_a4_c_2.pdf

Course Code	Course Name	L	T	P	Cr
DASH250052	Environmental Science	2	0	0	2

Course Outcomes:

CO1: Understand the importance and dimension of a healthy environment, become environmentally conscious, skilled and responsible in all their actions with a concern for sustainable development.
CO2: Comprehend the significance and issues related to ecosystems, natural resources and bio-diversity and become aware of the need and ways to protect/ preserve them.
CO3: Grasp the issues related to environmental pollution, solid waste management and climate change, and become conscious and proactive in the discharge of their responsibilities towards the environment.
CO4: Become aware and appreciate the values and concerns of environmental movements and policies and the role of communities, and act responsibly on environment-related issues.
CO5: Understand environmental policies, ethics, and community roles in sustainable development.

Module 1:

Introduction to Environmental Studies and Ecosystems:
Environmental Studies: Importance and scope, Multidisciplinary nature, Concept of sustainability and sustainable development, Ecosystems: Concept, Structure and Function, Pond and forest ecosystems, Food chains and food webs, Concept of ecological succession, Bio-geographical zones of India, Levels of biological diversity: Genetic, Species, and Ecosystem.

Module 2:

Biodiversity and Conservation:
Biodiversity Hotspots with special reference to India, Threats to biodiversity, Conservation of biodiversity: In-situ and Ex-situ, Endangered and endemic species: Concept Afforestation: Social forestry, Agroforestry, Green belt.

Module 3:

Pollution and Waste Management:
Air, water, and noise pollution: Causes, effects, and control measures, Climate change, global warming, ozone depletion, acid rain: Impact on humans and agriculture, Solid waste management: biodegradable and non-biodegradable waste.
- Segregation of domestic waste at source
- Impact of plastic on human and animal health

Module 4:

Natural Resources and Disaster Management:
Land resources and land-use changes, Land degradation, soil erosion, and desertification, Water: Use and over-exploitation, Water conservation : Rainwater harvesting, Watershed management : Meaning and importance, Energy resources: Renewable and non-renewable, Alternate energy sources, Disaster management – Types and self-protection during disasters Environmental Policy, 2006 : Provisions and importance; Environmental Impact Assessment – Concept; Swachh Bharat Mission Objectives; International agreements , Montreal and Kyoto protocols.

Module 5:

Environmental Policies and Human Role
Population growth: Impact on environment, health, and welfare, Environmental ethics : Religion and cultural role, Environmental movements: Chipko, Narmada Bachao, Silent Valley, Bishnoi's, Community initiatives:
Salu Marada Thimmakka, Sacred Groves (Devarakadu), Environment Protection Act, Biodiversity Act (2002), National Environmental Policy (2006), Environmental Impact Assessment, Swachh Bharat Mission, International agreements: Montreal and Kyoto Protocols.

Labs / Practicals:

NA

References:

1. Asthana, D. K. .Text Book of Environmental Studies. S. Chand Publishing.
2. Basu, M., Xavier, S. . Fundamentals of Environmental Studies, Cambridge University Press, India
3. Basu, R. N. Environment. University of Calcutta, Kolkata
4. Bharucha, E. Textbook of Environmental Studies for Undergraduate Courses. Universities Press, India.
5. De, A.K. Environmental Chemistry, New Age International, .
6. Mahapatra, R., Jeevan, S.S., Das, S. (Eds) . Environment Reader for Universities, Centre for Science and Environment.
7. Masters, G. M., &Ela, W. P. .Introduction to environmental engineering and science. Englewood Cliffs, NJ: Prentice Hall.
8. Odum, E. P., Odum, H. T., & Andrews, J. Fundamentals of ecology. Philadelphia: Saunders.
9. Sharma, P. D., & Sharma, P. D. .Ecology and environment. Rastogi Publications.

Course Code	Course Name	L	T	P	Cr
DCSE250107	Principles of Compiler Design	3	0	2	4

Course Outcomes:

CO1: Understand the various phases of a compiler and their roles in program translation.
CO2: Design and implement lexical analyzers and parsers using formal grammar and tools like LEX and YACC.
CO3: Apply syntax-directed translation techniques and manage symbol tables for semantic analysis.
CO4: Generate intermediate code and understand run-time environments and memory allocation strategies.
CO5: Analyze and apply code optimization techniques to improve the performance of the generated code.
CO6: Develop basic compilers or interpreters using compiler construction principles and tools

Module 1:

Introduction & Lexical Analysis: Overview of compilers: phases and architecture, Role and functions of lexical analyzer, Token specification, recognition, pattern matching, Finite automata: DFA, NFA, Tools: Lex/Flex and writing a hand crafted lexer

Module 2:

Syntax Analysis: Context Free Grammars (CFG), parse trees, ambiguity, Top down parsing: FIRST/FOLLOW, LL(1), recursive descent, Bottom up parsing: operator-precedence, LR(0), SLR(1), LR(1), LALR(1), Parser generators: YACC/Bison, error recovery strategies

Module 3:

Semantic Analysis & Syntax-Directed Translation: Syntax-Directed Definitions (SDDs): attributes, evaluation order, Types of SDDs: S attributed, L attributed, Semantic actions during parsing, Symbol tables, scope handling, type checking, semantic error handling

Module 4:

Intermediate Code Generation: Intermediate representations: three-address code, quadruples, triples, abstract syntax trees, SSA, Code translation: expressions, control flow, boolean logic, arrays, Backpatching, flow graphs, control flow translation

Module 5:

Runtime Environments & Code Generation: Activation records, stack and heap allocation, Accessing non-local data (nested procedures), Machine code generation: instruction selection, run time storage management, basic blocks
Register allocation strategies and graph coloring.
Code Optimization: Machine-independent optimizations: local and global (SSA, DAGs), Dataflow analysis, loop optimization, peephole techniques, Machine-dependent optimization fundamentals

Labs / Practicals:

1. Experiment No.1: Lexical Analyzer using LEX Tool: Design a lexical analyzer that identifies tokens such as identifiers, keywords, numbers, and operators.
2. Experiment No.2: Syntax Analyzer using YACC Tool: Implement a parser for arithmetic expressions using grammar rules in YACC and lexical rules in LEX.
3. Experiment No.3: Tokenization of Source Code (Manual or with LEX): Manually tokenize or use LEX to tokenize a given piece of C/C++ source code.
4. Experiment No.4: Implementation of a Recursive Descent Parser: Write a program for parsing a simple grammar using recursive descent parsing technique.
5. Experiment No.5: Construction of FIRST and FOLLOW Sets: Develop a program to compute the FIRST and FOLLOW sets for a given grammar.

6. Experiment No.6: Implementation of LL(1) Parsing Table: Construct the LL(1) parsing table from FIRST and FOLLOW sets and parse input strings accordingly.
7. Experiment No.7: Intermediate Code Generation (Three-Address Code): Generate three-address code (TAC) for assignment statements and arithmetic expressions.
8. Experiment No.8: Backpatching for Boolean Expressions and Control Structures: Simulate backpatching for control flow constructs like if, while, etc., during intermediate code generation.
9. Experiment No.9: Symbol Table Implementation: Implement a symbol table with insertion, lookup, and deletion operations for identifiers.
10. Experiment No.10: Simple Code Optimization Techniques: Apply basic optimization techniques such as constant folding, dead code elimination, or strength reduction on intermediate code.

References:

1. Aho, Sethi, Ullman: Compilers: Principles, Techniques & Tools (Pearson) studocu.com+4id.scribd.com+4en.wikipedia.org+4
2. Louden: Compiler Construction (Cengage) es.scribd.com+3scribd.com+3studocu.com+3
3. Appel: Modern Compiler Implementation
4. Tremblay & Sorrenson: Theory & Practice of Compiler Writing

Course Code	Course Name	L	T	P	Cr
DCSE250060	Database Management Systems	2	0	4	4

Course Outcomes:

CO1. Understand the basic database concepts, data models, schemas and instances.
CO2. Learn database design using Entity Relationship model and learn the use of constraints and relational algebra operations.
CO3. Gain proficiency in SQL and construct queries using SQL.
CO4. Understand the importance of normalization in database design.
CO5. Understand transaction processing, concurrency control and recovery

Module 1:

Introduction to Databases, File systems vs. Database systems, Characteristics of DBMS, Applications of DBMS, Advantages of using a DBMS, Database users and administrators, Three-level architecture of DBMS (ANSI/SPARC architecture), Data independence, Database schema, Instance, Generalization, Specialization, Structure of a DBMS, Overview of DBMS software (e.g., MySQL, SQL Server, Oracle, PostgreSQL)

Module 2:

Introduction to Data models: Hierarchical, Network, Relational, Object-oriented, Data Modeling using Entity Relationship (ER) model: entities, entity types, attributes, relationships, relationship types, E/R diagram notation, examples.
Relational data model: Concepts, schema, instances, keys, constraints, Relational algebra: Select, project, union, set difference, Cartesian product, joins, ER to Relational mapping, Relational algebra and relational calculus, Tuple Relational Calculus, Domain Relational Calculus

Module 3:

Structured Query Language (SQL): Basics of SQL, DDL, DML,DCL, structure – creation, alteration, defining constraints – Primary key, foreign key, unique, not null, check, IN operator, Functions - aggregate functions, Built-in functions – numeric, date, string functions, set operations, sub-queries, correlated sub-queries, Use of group by, having, order by, join and its types, Exist, Any, All , view and its types. Transaction control commands – Commit, Rollback, Save point, Cursors, Indexes, stored procedures, Triggers

Module 4:

Normalization – Introduction, Non loss decomposition and functional dependencies, First, Second, and third normal forms – dependency preservation, Boyce/Codd normal form.
Higher Normal Forms - Introduction, Multi-valued dependencies and Fourth normal form, Join dependencies and Fifth normal form.

Module 5:

Transaction management and Concurrency control: Transaction processing and Error recovery - concepts of transaction processing, ACID properties, and serializability concurrency control, Lock based concurrency control (2PL, Deadlocks), Time stamping methods, optimistic methods, and database recovery management. Error recovery and logging, undo, redo, undo-redo logging and recovery methods, Introduction to database security concepts.

Labs / Practicals:

1. Create a database, define tables using DDL, apply constraints (Primary, Foreign, Not Null, Unique).
2. Insert, update, delete, and retrieve records using SQL DML commands.
3. Write SQL queries using WHERE, ORDER BY, GROUP BY, HAVING, LIKE, etc.
4. Write SQL queries to demonstrate the use of INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, and SELF JOIN using appropriate sample tables.
5. Write nested subqueries with IN, EXISTS, and ANY/ALL.

6. Draw ER diagram for a sample application (e.g., Library or Hospital), map it to relational schema.
7. Normalize sample unstructured data up to 3NF or BCNF. Implement before-and-after schemas in SQL.
8. Create and execute stored procedures/functions for common operations.

References:

1. Silberschatz, Korth, Sudarshan – Database System Concepts, McGraw Hill Year: 2020 (7th edition).
2. Elmasri and Navathe – Fundamentals of Database Systems, Pearson Edition 2016
3. Raghu Ramakrishnan, Johannes Gehrke – Database Management Systems, McGraw Hill
4. Database Systems Concepts Author H.f.Korth and Silberschatz Publisher McGraw Hill
5. Database System Author Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom Publisher Pearson Edition 2nd Edition
6. C.J. Date – An Introduction to Database Systems, Pearson

Course Code	Course Name	L	T	P	Cr
DCSE250035	Computer Networks	3	0	2	4

Course Outcomes:

CO1: Understand Network Basics – Learn network models, topologies, and transmission media.
CO2: Analyse Data Link Layer – Evaluate error detection, MAC protocols, and wireless communication.
CO3: Implement Network Layer Functions – Apply IP addressing, subnetting, and routing algorithms.
CO4: Apply Transport & Application Layer Concepts – Compare TCP/UDP and analyse key protocols.
CO5: Demonstrate Security & Emerging Technologies – Identify threats, implement cryptography, explore SDN, IoT, and cloud networking.

Module 1:

Introduction to Computer Networks Overview of Computer Networks, Network Types: LAN, WAN, MAN, PAN, Network Topologies: Bus, Star, Ring, Mesh, Hybrid, OSI Model & TCP/IP Model – Layers and Functions, Transmission Media: Wired & Wireless Technologies, Switching Techniques: Circuit, Packet, and Message Switching

Module 2:

Data Link Layer & MAC Protocols
Error Detection and Correction: Parity, Checksum, CRC, Hamming Code, Flow Control Mechanisms: Stop-and-Wait, Sliding Window Protocol, MAC Protocols: ALOHA, CSMA/CD, CSMA/CA, Ethernet Standards (IEEE 802.3), VLANs, Spanning Tree Protocol, Wireless LAN (Wi-Fi), Bluetooth, and Mobile Networks

Module 3:

Network Layer & Routing Algorithms
IPv4 and IPv6 Addressing, Subnetting, and CIDR, Routing Algorithms: Distance Vector, Link-State, and Hybrid Routing, Routing Protocols: RIP, OSPF, EIGRP, and BGP, Congestion Control Mechanisms and Quality of Service (QoS), Network Address Translation (NAT) and DHCP

Module 4:

Transport Layer & Application Layer Protocols
TCP and UDP: Connection Establishment, Flow Control, and Error Control, Congestion Control: TCP Reno, TCP Tahoe, and Fair Queuing, Application Layer Protocols: HTTP, HTTPS, FTP, SMTP, POP3, IMAP, DNS and URL Resolution, Peer-to-Peer Networks, Security in Transport & Application Layers: TLS, SSL, Firewalls

Module 5:

Network Security and Emerging Trends
Cryptography Fundamentals: Symmetric and Asymmetric Encryption, Authentication and Digital Signatures, Network Security Attacks: DoS, DDoS, Phishing, Man-in-the-Middle, Wireless and IoT Security Challenges, Software-Defined Networking (SDN) and Cloud Networking, Introduction to 5G, Edge Computing, and Quantum Networking

Labs / Practicals:

Preparing

References:

1. "Computer Networking: A Top Down Approach (Eighth Edition)" by James F. Kurose and Keith W. Ross

https://gaia.cs.umass.edu/kurose_ross/about.php

2. "An Introduction to Computer Networks (Second Edition)" by Peter Lars Dordal
https://intronetworks.cs.luc.edu/current2/ComputerNetworks.pdf

3. "Computer Networking: Principles, Protocols and Practice (Third Edition)" by Olivier Bonaventure
https://github.com/cnp3/ebook/releases/download/draft-3rd/CNP3-2021.pdf

4. "Computer Networks: A Systems Approach (Sixth Edition)" by Larry L. Peterson and Bruce S. Davie
https://github.com/SystemsApproach/book/releases/download/v6.1/book.pdf

5. "Computer Networks (Fifth Edition)" by Andrew Stuart Tanenbaum and David J. Wetherall
https://www.oreilly.com/library/view/computer-networks-fifth/9780133485936

6. "Internetworking with TCP/IP: Principles, Protocols, and Architecture (Sixth Edition)" by Douglas Earl Comer
https://www.oreilly.com/library/view/internetworking-with-tcpip/9780137464197

7. "UNIX Network Programming: Volume 1 (Third Edition)" by William Richard Stevens, Bill Fenner, and Andrew M. Rudoff
https://www.oreilly.com/library/view/the-sockets-networking/0131411551

8. "Computer and Network Organization: An Introduction" by Maarten van Steen and Henk Sips
https://www.distributed-systems.net/index.php/books/computer-and-network-organization

9. "Beej's Guide to Network Programming" by Brian Hall
https://beej.us/guide/bgnet/pdf/bgnet_a4_c_2.pdf

Course Code	Course Name	L	T	P	Cr
DASH250026	Design Thinking	1	0	2	2

Course Outcomes:

CO1 : Understand the core principles and process of design thinking including empathy, ideation, and prototyping.
CO2 : Apply user-centric research methods to identify and frame real-world problems.
CO3 : Generate and evaluate innovative ideas through brainstorming and collaborative design activities.
CO4 : Build and test low-fidelity prototypes to explore solution possibilities.
CO5 : Present and communicate design solutions effectively using appropriate tools and storytelling techniques.

Module 1:

Introduction to Design Thinking
Definition and Importance of Design Thinking, Comparison with Traditional Problem Solving, Mindsets for Innovation, Stanford d.school Design Process, Applications in Engineering, Business, and Social Sectors.

Module 2:

Empathize – Understanding Users
Importance of Empathy, Observation and Interview Techniques, User Persona Creation, Empathy Mapping, Journey Maps, Capturing Insights from Field Research.

Module 3:

Define and Ideate
Problem Definition and Framing: Point of View (POV) Statements, Ideation Techniques (Brainstorming, SCAMPER, Mind Mapping), Idea Evaluation and Selection Criteria.

Module 4:

Prototype and Test
Prototyping Tools and Techniques, Rapid Prototyping, Paper Prototypes, Role-Playing, Usability Testing, Feedback Integration and Iteration.

Module 5:

Communication and Implementation
Storyboarding, Pitching Ideas, Design Thinking for Entrepreneurship, Real-life Case Studies, Ethical and Sustainable Design Considerations.

Labs / Practicals:

1. Conduct user interviews and observations to develop empathy maps.
2. Create user personas and problem statements.
3. Brainstorm solutions for a chosen problem and organize ideas using mind maps.
4. Develop low-fidelity prototypes for the selected ideas.
5. Test prototypes with users and gather feedback for refinement.
6. Present project using storyboards and pitch decks.
7. Group project: Apply the design thinking process to a real-world challenge and document the journey.
8. Use tools like Canva, Miro, or Figma for visualizing and presenting ideas.
9. Role-play-based usability tests.
10. Peer review and reflection journal on the design process.

References:

1. Tim Brown, Change by Design, Harper Business.
2. Rolf Faste, Design Thinking Methodology, Stanford University Materials.
3. Jeanne Liedtka & Tim Ogilvie, Designing for Growth: A Design Thinking Toolkit for Managers, Columbia

Business School Publishing.
4. Nigel Cross, Design Thinking: Understanding How Designers Think and Work, Berg Publishers.
5. Plattner, Meinel, Leifer, Design Thinking: Understand – Improve – Apply, Springer.

Course Code	Course Name	L	T	P	Cr
DCSA250056	Numerical Methods	3	1	0	4

Course Outcomes:

CO1: Solve algebraic and transcendental equations using appropriate numerical methods.
CO2: Apply interpolation techniques to estimate values for given discrete data.
CO3: Use numerical techniques to solve systems of linear equations efficiently.
CO4: Compute derivatives and definite integrals using numerical differentiation and integration methods.
CO5: Solve ordinary differential equations using numerical approaches and analyze the accuracy of solutions.

Module 1:

Solution of Equations and Interpolation:
Bisection method, Regula-Falsi method, Newton-Raphson method, Forward & Backward differences, Newton's & Gauss interpolation, Lagrange interpolation, Divided difference method

Module 2:

Numerical Differentiation and Integration:
Numerical differentiation using finite differences
Numerical integration using: Trapezoidal rule, Simpson's 1/3 rule, Simpson's 3/8 rule

Module 3:

Solution of Ordinary Differential Equations (ODEs):
Taylor series method, Euler's method, Modified Euler's method, Runge-Kutta methods (2nd and 4th order)

Module 4:

Predictor-Corrector Methods:
Milne's predictor-corrector method, Adams-Bashforth and Adams-Moulton methods

Module 5:

Numerical Solution of Partial Differential Equations (PDEs):
Finite difference method for Laplace and Poisson equations, Heat equation (Explicit and Crank-Nicholson methods), Wave equation using explicit method

Labs / Practicals:

N/A

References:

1. P. Kandasamy, K. Thilagavathy & K. Gunavathi, Numerical Methods (S. Chand, 2nd Ed, 2012)
2. S. S. Sastry, Introductory Methods of Numerical Analysis (PHI, 4th Ed, 2005)
3. E. Kreyszig, Advanced Engineering Mathematics (Wiley, 9th Ed, 2006)
4. B. S. Grewal, Higher Engineering Mathematics (Khanna, 35th Ed, 2010

Course Code	Course Name	L	T	P	Cr
DCSE250082	Game Theory and Mechanism Design	3	1	0	4

Course Outcomes:

CO1. Understand the fundamental concepts and classifications of games.
CO2. Analyze strategic interactions using solution concepts like Nash equilibrium.
CO3. Apply game-theoretic principles to real-world competitive and cooperative scenarios.
CO4. Understand the design and properties of mechanisms and incentive structures.
CO5. Evaluate applications of mechanism design in economics, auctions, and distributed systems.

Module 1:

Introduction to Game Theory:
Classification of games – cooperative vs non-cooperative, static vs dynamic, complete vs incomplete information; Representation of games – normal form and extensive form; Rationality, preferences, utility functions; Dominant and dominated strategies.

Module 2:

Solution Concepts in Non-Cooperative Games:
Nash equilibrium – existence and uniqueness, best response strategies; Mixed strategy equilibria; Prisoner's dilemma, battle of the sexes, matching pennies; Iterated elimination of dominated strategies; Applications in routing, load balancing, and resource allocation.

Module 3:

Games with Incomplete Information and Repeated Games:
Bayesian games, Bayesian Nash equilibrium, signaling and screening; Repeated games – finitely and infinitely repeated games, strategies in repeated games, Folk theorem, punishment strategies, reputation effects; Case studies in trust and cooperation.

Module 4:

Cooperative Game Theory:
Coalitional games – core, Shapley value, bargaining solutions, Nash bargaining; Transferable vs non-transferable utility games; Stable matching, the Gale-Shapley algorithm; Applications in group decision making, voting systems, and cost sharing.

Module 5:

Mechanism Design and Applications:
Basics of mechanism design – incentive compatibility, revelation principle, dominant strategy incentive compatibility (DSIC), social choice functions; Vickrey-Clarke-Groves (VCG) mechanism, auctions – first price, second price, combinatorial auctions; Applications in e-commerce, resource allocation, and public goods provision.

Labs / Practicals:

NA

References:

1. Martin J. Osborne – An Introduction to Game Theory, Oxford University Press
2. Vijay Krishna – Auction Theory, Academic Press
3. Y. Narahari – Game Theory and Mechanism Design, IISc Press and World Scientific
4. Drew Fudenberg and Jean Tirole – Game Theory, MIT Press
5. Roger B. Myerson – Game Theory: Analysis of Conflict, Harvard University Press

Course Code	Course Name	L	T	P	Cr
DCSE250043	Cryptography and Network Security	3	0	2	4

Course Outcomes:

CO1: Explain security fundamentals, including the CIA Triad and various cyber threats.
CO2: Describe security services and mechanisms, such as authentication and encryption.
CO3: Understand the history and evolution of cryptography, including classical techniques.
CO4: Recognize the role of cryptanalysis in breaking cryptographic systems.
CO5: Apply mathematical foundations, including number theory, modular arithmetic, and random number generation, in cryptographic algorithms.

Module 1:

Introduction to Cryptography & Security Concepts
Basics of Security, Security Goals: Confidentiality, Integrity, Availability (CIA), Threats and Attacks: Passive vs. Active Attacks, Security Services & Mechanisms, Cybersecurity Principles, Introduction to Cryptography, History & Evolution of Cryptography, Classical Cryptographic Techniques: Caesar Cipher, Vigenère Cipher, Playfair Cipher, Modern Cryptography & Cryptanalysis
Mathematical Foundations of Cryptography: Number Theory Basics: Prime Numbers, Modular Arithmetic, Euler's Theorem, Greatest Common Divisor (GCD), Fermat's Theorem, Random Number Generation

Module 2:

Symmetric & Asymmetric Cryptography
Symmetric Key Cryptography, Block Ciphers vs. Stream Ciphers, Data Encryption Standard (DES) & Triple DES, Advanced Encryption Standard (AES), Modes of Operation: ECB, CBC, CFB, OFB, CTR, Asymmetric Key Cryptography, Public Key Cryptosystems: RSA Algorithm, Key Exchange Mechanisms: Diffie-Hellman Key Exchange, Elliptic Curve Cryptography (ECC), Comparison of Symmetric & Asymmetric Cryptography

Module 3:

Hash Functions, Digital Signatures & Authentication Cryptographic Hash Functions, Properties of Hash Functions, Common Hash Algorithms: MD5, SHA-1, SHA-256, SHA-3, Message Authentication, Message Authentication Codes (MAC), HMAC (Hash-based Message Authentication Code), Digital Signatures & Certificates, RSA Digital Signature Algorithm, Digital Certificates & Public Key Infrastructure (PKI), SSL/TLS Encryption and HTTPS Authentication Mechanisms, Password-Based Authentication, Multi-Factor Authentication (MFA), Biometric Authentication

Module 4:

Network Security & Secure Protocols
Network Security Fundamentals, Security in OSI & TCP/IP Models, Firewalls and Intrusion Detection Systems (IDS/IPS), Virtual Private Networks (VPNs) & IPsec, Secure Network Protocols, Secure Email: PGP & S/MIME, Secure Web Transactions: HTTPS, SSL/TLS
Wireless Security: WPA, WPA2, WPA3, Network Attacks & Defense Mechanisms, Denial of Service (DoS) & Distributed DoS (DDoS) Attacks, Man-in-the-Middle (MITM) & Eavesdropping, SQL Injection & Cross-Site Scripting (XSS)

Module 5:

Emerging Trends & Cybersecurity Applications
Blockchain & Cryptography, Role of Cryptography in Blockchain, Bitcoin & Ethereum Security, Cloud Security & Zero Trust Security Model, Cloud Encryption Mechanisms, Zero Trust Architecture & Access Control
Cybersecurity & Ethical Hacking, Introduction to Ethical Hacking, Penetration Testing Basics, Legal & Ethical

Issues in Cybersecurity, Future Trends in Cryptography & Security

Labs / Practicals:

Preparing

References:

1. "The Joy of Cryptography" by Michael Rosulek
https://joyofcryptography.com/pdf/book.pdf

2. "Cryptography and Computer Security" by Chris Bourke
https://cse.unl.edu/~cbourke/cryptoNotes.pdf

3. "Applied Cryptography (20th Anniversary Edition)" by Bruce Schneier
https://www.schneier.com/books/applied-cryptography

4. "Cryptography and Network Security (Eighth Edition)" by William Stallings
http://williamstallings.com/Cryptography/Crypto8e-Student

5. "Cracking Codes With Python: An Introduction To Building And Breaking Ciphers" by Al Sweigart
https://inventwithpython.com/cracking

6. "Yet Another Introductory Number Theory Textbook (Cryptology Emphasis Version)" by Jonathan Adam Poritz
https://www.poritz.net/jonathan/share/yaintt.pdf

Course Code	Course Name	L	T	P	Cr
DASH250056	Financial Accounting & Management	2	0	0	2

Course Outcomes:

CO1: Describe fundamental accounting principles and management concepts.
CO2: Prepare basic accounting statements such as journal entries, ledger, and trial balance.
CO3: Analyze financial statements using ratio analysis and cost analysis techniques.
CO4: Explain budgeting, working capital management, and financial planning in organizations.
CO5: Apply financial decision-making concepts in real-world business or startup contexts.

Module 1:

Introduction to Accounting and Management:
Need and scope of accounting, principles and concepts of accounting, objectives and functions of management, differences between accounting and management, users of accounting information, relevance of financial literacy for computer professionals and entrepreneurs.

Module 2:

Accounting Process:
Double-entry system of accounting, journal entries, posting to ledger; preparation of trial balance, understanding of cash book and petty cash book, simple problems on journal and ledger for service and product-based businesses.

Module 3:

Financial Statements and Analysis:
Preparation of trading account, profit and loss account, and balance sheet (with adjustments), Basics of financial statement interpretation, Ratio analysis: liquidity, profitability, solvency, turnover ratios, Use of spreadsheet tools (Excel) for analysis.

Module 4:

Cost and Budgetary Control:
Types of costs (fixed, variable, direct, indirect), marginal costing and break-even analysis, preparation of budgets: cash, production, flexible, budgetary control techniques, basics of working capital management.

Module 5:

Financial Management and ICT Integration:
Time value of money, capital budgeting basics, sources of finance, digital tools and ERPs (Tally, Zoho, SAP basics), GST basics and e-invoicing, case studies on fintech, budgeting in startups, digital payment ecosystems in India.

Labs / Practicals:

N/A

References:

1.S.N. Maheshwari & S.K. Maheshwari – An Introduction to Accountancy, Vikas Publishing.
2.T.S. Grewal – Double Entry Book Keeping, Sultan Chand & Sons.
3.M.Y. Khan & P.K. Jain – Financial Management: Text, Problems and Cases, McGraw Hill.
4.R.P. Rustagi – Fundamentals of Financial Management, Taxmann Publications.
5.Dr. Jawahar Lal – Accounting for Management, Himalaya Publishing.
6.NCERT Bookkeeping and Accountancy (Class XI & XII) – Foundation Concepts.

Course Code	Course Name	L	T	P	Cr
DASH250105	Universal Human Values-II: Understanding Harmony And Ethical Human Conduct	3	0	0	3

Course Outcomes:

CO1: Understand human aspirations and distinguish between needs of the self and the body.
CO2: Analyze the concept of harmony in self, family, society, and nature.
CO3: Apply ethical principles to real-life situations and emerging challenges in science and technology.
CO4: Evaluate responsible behavior in professional and personal domains with sustainability and justice in mind.
CO5: Develop a holistic perspective for ethical human conduct in the context of AI, data use, and digital society.

Module 1:

Introduction to Value Education and Human Aspirations
Purpose of education, the role of values in personal and professional life; Understanding happiness and prosperity; Distinction between needs of the Self ('I') and Body; right understanding and right feelings; AI ethics and human well-being.

Module 2:

Harmony in the Human Being – Understanding the Self
Self-exploration as the process for value education; Understanding the activities of the Self and harmony in them; Program to ensure harmony; Applying this to deal with digital distractions, identity crisis, and mental well-being.

Module 3:

Harmony in Family and Society
Values in human relationships – trust and respect; Nine universal feelings in relationships; harmony in society through justice, mutual fulfillment, and participation; Ethical dilemmas in teamwork, AI product development, and organizational conduct.

Module 4:

Harmony with Nature and Sustainable Living
Interconnectedness of human and natural systems; Four orders of nature; Co-existence and Eco-balance; Sustainable production, consumption, and innovation in AI/ML; Responsible use of computing resources and data centers.

Module 5:

Ethical Human Conduct and Universal Order
Universal human order (Sarvabhauma Vyavastha); Ethical competence; vision for a humane society; Professional ethics for technologists – data privacy, bias in AI, ethical design of systems; role of engineers in promoting equity, well-being, and digital justice.

Labs / Practicals:

N/A

References:

Textbooks & Guides:
1. R.R. Gaur, R. Sangal, G.P. Bagaria – A Foundation Course in Human Values and Professional Ethics, Excel Books (AICTE
Recommended)
2. Ritu Sinha – Understanding Human Values, Laxmi Publications
3. B. L. Bajpai – Indian Ethos and Modern Management, New Royal Book Co.
4. R. Subramanian – Professional Ethics, Oxford University Press India
5. K. Mohan & M. Banerjee – Developing Soft Skills and Personality, McGraw Hill India

Course Code	Course Name	L	T	P	Cr
DCSE250004	Advanced Data Structures and Algorithms	3	0	2	4

Course Outcomes:

CO1: Analyze the performance of basic data structures and sorting/searching techniques.
CO2: Implement and apply advanced trees, heaps, and hash tables to solve complex problems.
CO3: Solve problems using graph algorithms and greedy strategies.
CO4: Design algorithms using divide and conquer, backtracking, and dynamic programming.
CO5: Develop efficient and scalable solutions using appropriate algorithmic techniques.

Module 1:

Review of Fundamental Concepts
Pointers and Dynamic Memory Allocation, Arrays, Dynamically Allocated Arrays, Structures, Algorithm Specification, Data Abstraction, Performance Analysis, Performance Measurement, Time and space complexity

Module 2:

Basic Data Structures and Advanced Trees
Stacks, Queues, Linked Lists, Hash Tables, Searching and Sorting Techniques: Linear Search, Bubble Sort, Selection Sort, Insertion Sort, Binary Trees, Binary Tree Traversals, Applications of Binary Trees, Threaded Binary Trees, Binary Search Trees, Advanced Trees: AVL Trees: Rotations, insertion and deletion, Red-Black Trees, Splay Trees, B Trees, B+ Trees, Binary Heap: Min/Max heap operations, Priority Queue applications.

Module 3:

Divide and Conquer, Backtracking, String Matching algorithms
Merge Sort, Quick Sort, Binary Search, Backtracking: N-Queens, Graph Coloring, String matching algorithms: KMP, Rabin-Karp

Module 4:

Graph Algorithms
Graph Representations: Adjacency list, Adjacency Matrix, BFS, DFS, Shortest Path: Dijkstra’s, Greedy algorithms: Activity Selection, Huffman Coding Minimum Spanning Tree: Prim’s and Kruskal’s Algorithms

Module 5:

Dynamic Programming
Basic concepts: Overlapping subproblems, optimal substructure, 0/1 Knapsack, Longest Common Subsequence, Matrix Chain Multiplication

Labs / Practicals:

Preparing

References:

1. "Introduction to Algorithms (Fourth Edition)" by Thomas H. Cormen, Charles Eric Leiserson, Ronald Linn Rivest, Clifford Seth Stein
https://mitpress.mit.edu/9780262046305/introduction-to-algorithms

2. "A Second Course in Algorithms" by Timothy Avelin Roughgarden
https://timroughgarden.org/w16/l

3. "Algorithm Design" by Jon Michael Kleinberg and Eva Tardos
https://dl.acm.org/doi/10.5555/1051910

4. "Computational Intractability: A Guide to Algorithmic Lower Bounds" by Erik D. Demaine, William Ian Gasarch, and Mohammad Taghi Hajiaghayi
https://hardness.mit.edu/drafts/2025-02-02.pdf

5. "The Design and Analysis of Algorithms" by Dexter Campbell Kozen
https://www.cs.cornell.edu/kozen/Papers/daa.pdf

6. "The Art of Computer Programming" by Donald Ervin Knuth
https://www-cs-faculty.stanford.edu/~knuth/taocp.html

Course Code	Course Name	L	T	P	Cr
DCSA250071	Software Engineering	3	1	0	4

Course Outcomes:

CO1: Understand Software Engineering Concepts – Explain core principles, ethical responsibilities, software processes, and critical system properties.
CO2: Analyze Software Requirements – Differentiate functional and non-functional requirements and apply requirements engineering processes.
CO3: Apply System Modeling & Project Management – Develop system models and manage software projects, including planning, scheduling, and risk management.
CO4: Design & Develop Software – Implement architectural and object-oriented design, agile methodologies, and software evolution strategies.
CO5: Perform Software Testing & Cost Estimation – Conduct verification, validation, software testing, team management, and cost estimation techniques

Module 1:

Overview - Introduction, FAQs about software engineering, Professional and ethical responsibility; Socio-technical systems - Emergent system properties, Systems engineering; Critical systems and Software processes - Critical systems, A simple safety-critical system, System dependability, Availability and reliability; Software processes - Software process models, Process iteration, Process activities, The Rational Unified Process, Computer-Aided Software Engineering.

Module 2:

Requirements - Software requirements, Functional and non-functional requirements, User requirements, System requirements, Interface specification, The software requirements document; Requirements engineering processes - Feasibility studies, Requirements elicitation and analysis, Requirements validation, Requirements management.

Module 3:

System models and Project Management - System models, Context models, Behavioural models, Data models, Object models, Structured methods; Project management - Management activities, Project planning, Project scheduling, Risk management

Module 4:

Software Design - Architectural design, Architectural design decisions, System organisation, Modular decomposition styles, Control styles; Object-oriented design - Objects and object classes, An object-oriented design process, Design evolution; Development - Rapid software development, Agile methods, Extreme programming, Rapid application development; Software evolution - Program evolution dynamics, Software maintenance, Evolution processes, Legacy system evolution.

Module 5:

Verification and validation - Verification and validation, Planning verification and validation, Software inspections, Automated static analysis, Verification and formal methods; Software testing - System testing, Component testing, Test case design, Test automation; Management - Managing people, Selecting staff, Motivating people, Managing groups, The People Capability Maturity Model; Software cost estimation, Software productivity, Estimation techniques, Algorithmic cost modelling, Project duration and staffing

Labs / Practicals:

N/A

References:

1.Ian Sommerville: Software Engineering, 8th Edition, Pearson Education, 2007.
2.Roger S. Pressman: Software Engineering - A Practitioner’s Approach, 7th Edition, Tata McGraw Hill, 2007.
3.Pankaj Jalote: An Integrated Approach to Software Engineering, Wiley India, 2009.

Course Code	Course Name	L	T	P	Cr
DCSE250073	Ethical Hacking	2	1	2	4

Course Outcomes:

CO1: Explain the ethical hacking process and distinguish between various attack vectors, hacker types, and phases of penetration testing.
CO2: Apply reconnaissance techniques such as footprinting, scanning, and enumeration to gather information about target systems.
CO3: Demonstrate the ability to exploit system and network vulnerabilities using tools and techniques like password cracking, sniffing, and privilege escalation.
CO4: Identify and exploit web application and wireless network vulnerabilities using ethical hacking tools and analyze the associated risks.
CO5: Prepare and document penetration testing reports while adhering to legal and ethical standards of cybersecurity practice.

Module 1:

Ethical Hacking Process & Reconnaissance: Ethical hacking methodology
Hacker mindset, anonymity, legality, ethics, Information gathering, techniques (active & passive), Physical security weaknesses, Internal & external testing, Penetration test planning and reporting

Module 2:

Social Engineering & Password Attacks: Types of social engineering, Phishing, pretexting, tailgating, Password cracking techniques (brute force, dictionary), Privilege escalation, Countermeasures

Module 3:

Sniffing & Network Scanning: Packet sniffing methods, ARP spoofing, DNS/IP sniffing, Network scanning tools and techniques

Module 4:

System and Wireless Hacking: System hacking: vulnerability discovery, keystroke logging, remote exploits, Wireless hacking: WEP/WPA weakness, wireless traffic capturing, cracking

Module 5:

Web & Application Hacking: Web-based attacks: SQL Injection, XSS, file inclusion, Use of tools like Nikto, Burp Suite, Metasploit, Wireless, Web app vulnerability exploitation and reporting

Labs / Practicals:

1. Experiment No.1: Footprinting and Reconnaissance: Perform passive and active information gathering on a target website or IP using tools like whois, nslookup, the Harvester, and Recon-ng.
2. Experiment No.2: Scanning Networks: Use Nmap or Zenmap to discover open ports, services, and operating systems on a target machine.
3. Experiment No.3: Enumeration: Enumerate usernames, shared resources, and services using tools like NetBIOS, SNMPwalk, or enum4linux.
4. Experiment No.4: System Hacking – Password Cracking: Perform password cracking using tools like John the Ripper, Hydra, or Hashcat.
5. Experiment No.5: Malware Analysis (Static and Dynamic): Analyze a sample malware in a sandbox environment using tools like PEStudio, Process Monitor, or Wireshark.
6. Experiment No.6: Sniffing and Spoofing: Capture and analyze network packets using Wireshark or tcpdump; perform ARP spoofing using Ettercap or Cain and Abel.
7. Experiment No.7: Session Hijacking: Demonstrate session hijacking using tools like Ettercap or browser

cookie stealing techniques in a controlled environment.
8. Experiment No.8: Denial of Service (DoS) Attack: Simulate a DoS attack using tools like Hping3, LOIC, or custom scripts to understand impact and prevention.
9. Experiment No.9: Web Server Footprinting and Vulnerability Scanning: Perform vulnerability scanning on a web server using tools like Nikto, OpenVAS, or Nessus.
10. Experiment No.10: SQL Injection Attack: Demonstrate SQL injection on a test web application using tools like sqlmap or manual techniques.
11. Experiment No11: Cross-Site Scripting (XSS) Attack: Perform stored and reflected XSS attacks on a vulnerable web application like DVWA (Damn Vulnerable Web Application).
12. Experiment No.12: Cross-Site Request Forgery (CSRF) Attack: Simulate CSRF attacks using web applications like DVWA or custom scripts to understand defense mechanisms.
13. Experiment No.13: Privilege Escalation: Perform privilege escalation in a Linux or Windows environment using local exploits or misconfigurations.
14. Experiment No.14: Wireless Hacking: Crack WPA/WPA2 passwords using aircrack-ng suite and perform wireless network analysis.
15. Experiment No.15: Creating a Penetration Testing Report: Conduct a full penetration test on a test system or network and document findings, tools used, vulnerabilities, and mitigation strategies.

References:

1. Harris, S., Harper, A., Eagle, C., & Ness, J. (Year). Gray Hat Hacking: The Ethical Hacker’s Handbook. TMH.
2. Erickson, J. (Year). Hacking: The Art of Exploitation. SPD.
3. Beaver, K. (2013). Hacking for Dummies (3rd ed.). Wiley.
4. Additional: Shon Harris et al., Gray Hat Hacking (TMH); Thomas Mathew, Ethical Hacking (OSB).

Course Code	Course Name	L	T	P	Cr
DCSA250031	Full Stack Web Development	3	0	2	4

Course Outcomes:

CO1: Explain full stack development concepts, client-server model, and use essential development tools effectively.
CO2: Develop interactive web pages using HTML, CSS, JavaScript events, form validation, and dynamic content.
CO3: Build React applications with components, JSX, props, state, interactive forms, and conditional rendering.
CO4: Create backend APIs using Node.js and Express, connect with frontend, and exchange JSON data.
CO5: Integrate frontend and backend features into a functional mini project with data persistence.

Module 1:

What is full stack development? Difference between front-end and back-end, Understanding how websites work (client/server model),Introduction to tools: VS Code, Chrome, Node.js, GitHub.

Module 2:

Simple web page structure using HTML tags, CSS for styling layout and elements JavaScript basics: variables, events, interaction, Handling user actions like clicks and input, Creating and validating simple forms, Dynamic content display using JavaScript.

Module 3:

Introduction to React and benefits, Setting up a React project, Understanding components and JSX, Passing data with props and using useState, Creating interactive forms, /Handling input changes and button clicks, Rendering lists and conditional content.

Module 4:

Introduction to Backend and APIs, What is Node.js? What is a server? Creating a simple Express serve, Sending and receiving JSON. Simple REST API with Express, Create routes: GET, POST, Connecting Backend to Frontend: Calling backend from React using fetch, Displaying backend data in UI, Submitting form data to backend.

Module 5:

Developing a mini project Using the Features: Frontend form (React), Backend data saving (Node), displaying data list.

Labs / Practicals:

preparing

References:

1. "Full Stack Development with React and Node.js",By David Choi
2. "Pro MERN Stack" (2nd Edition),By Vasan Subramanian
3. JavaScript and JQuery: Interactive Front-End Web Development",By Jon Duckett
4. "Learning Full-Stack JavaScript Development",By Eric Bush

Course Code	Course Name	L	T	P	Cr
DOAI260001	AI (Industry)-Applied Ethics for AI	3	0	2	4

Course Outcomes:

CO1. Understand and be able to define and explain the terms 'AI for good' and 'responsible AI'. Describe the significance of responsible AI and recognize the potential benefits of AI in society.
CO2. Discuss and examine the impact of ethics on AI decision-making processes. Identify common ethical pitfalls in AI development and deployment. Explore methods for ethical AI practices.
CO3. Understand to apply an ethics-first approach to AI project planning and development. Compare different approaches to ethical AI design which include design of rules based on ethics and virtues. Identify factors influencing ethical decisions in AI development.
CO4. Apply Utilitarian and Kantian ethical principles to AI case studies, scenarios, and dilemmas. Explain the purpose and benefits of AI ethics codes. Develop a code of ethics for AI within an organization.
CO5. Identify the importance of interpretability in AI models. Evaluate the strengths and weaknesses of three different explainable AI tools: SHAP, LIME, and Tree Interpreter.
CO6. Conduct impact assessments for AI solutions. Apply ethical considerations in designing an AI system or solution. Implement measures to mitigate ethical risks and challenges post deployment.

Module 1:

Introduction to AI Ethics – meaning and importance, Role of Ethics in Responsible AI – defining 'AI for good' and 'responsible AI', significance of responsible AI in society, Common Ethical Pitfalls in AI – bias, fairness, accountability, transparency issues in AI development and deployment, Principles of Ethical AI – core principles guiding ethical AI, methods for ethical AI practices in development and deployment.

Module 2:

Ethics-First Approach in the AI Project Cycle – applying ethics at every stage of AI project planning and development, Approaches for Designing Ethical AI – rules-based ethics, values-based design, Introduction to Ethical Frameworks – overview of major ethical frameworks applied to AI, Virtue-Based Ethics – virtue ethics principles and the 4-box method for analysing ethical dilemmas, Applying Bio-Ethics Framework in AI – bioethical principles applied to AI in healthcare and biotechnology.

Module 3:

Utilitarianism for Ethical AI – utilitarian principles applied to AI case studies and dilemmas, Kantianism for Ethical AI – Kantian ethical principles applied to AI scenarios, Code of AI Ethics for Organizations – purpose, benefits and development of an organizational AI code of ethics, Data Protection and AI – risks of data breaches, privacy violations, responsible data handling.

Module 4:

Explainable AI – importance of interpretability in AI models, Intel XAI Toolkit, impact of explainability on decision-making and trust, Explainable AI in Practice – SHAP, LIME and Tree Interpreter tools: strengths, weaknesses and methods for applying them to analyze and interpret AI applications.

Module 5:

Impact Assessment of AI Solutions – conducting impact assessments for AI systems, Mitigating Ethical Issues Post Deployment – strategies to identify and resolve ethical risks after deployment, Project Creation – creating an AI startup with data card, model card, code of ethics, explainability report, impact self-assessment and transparency report.

Labs / Practicals:

1. Develop comprehensive Data Cards for datasets, as outlined in the Data Cards Playbook by Google, to ensure transparency and accountability in AI training data.
2. Design Responsible AI Best Practices.

3. Apply Human-Centered AI Canvas to design AI systems that address user needs and ethical considerations.
4. Learn to use The Box to embed AI principles into organizational workflows and operations.
5. Apply bio-ethics principles to AI case studies, scenarios, and dilemmas; evaluate the impact of virtue-based ethics on AI systems.
6. Develop AI ethics cards for given AI scenarios. Analyse a variety of ethical dilemmas by following the 4-box method from Virtue Ethics.
7. Analyse a variety of ethical dilemmas by walking through the ethical frameworks: Virtue Ethics, Bioethics, Utilitarian, Kantian, and Moral System Agnostic.
8. Develop an AI Code of Ethics for a fictitious company.
9. Use the UXAI Toolkit to brainstorm and plan different explainable interfaces for AI applications. Use the explainable AI library SHAP to create an explainability report for income prediction on the standard adult census income dataset.
10. Conduct impact assessments for AI solutions; create your own AI startup, develop business documents, write documentation (data card, model card, code of ethics, explainability report, impact self-assessment) and create and analyze transparency report.

References:

1. Intel AI for Future Workforce – Applied Ethics for AI Course Content, Intel Digital Readiness Program.
2. Virginia Eubanks – Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor, St. Martin's Press.
3. Cathy O'Neil – Weapons of Math Destruction, Crown Publishers.
4. Kate Crawford – Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence, Yale University Press.
5. Google Data Cards Playbook – https://pair-code.github.io/datacardsplaybook/

Course Code	Course Name	L	T	P	Cr
DOAI260002	AI (Industry)-Introduction to Artificial Intelligence	3	0	2	4

Course Outcomes:

CO1. Describe what AI is and what it is not. Appreciate the history of AI and its evolution over time.
CO2. Identify and analyze current trends in AI by correlating them with other technologies like IoT, Big Data and 5G.
CO3. Identify 3 common domains of AI – Natural Language Processing, Computer Vision and Statistical Data – based on the type of underlying data.
CO4. Examine and appreciate typical steps involved in an AI Project through the AI Project Cycle.
CO5. Examine the immediate impacts of AI practices on society and appreciate the ethical concerns of AI applications.
CO6. Classify Supervised Learning, Unsupervised Learning and Reinforcement Learning with the help of examples. Discuss the roles played by different key stakeholders in a typical AI team.

Module 1:

Demystification of AI – what AI is and what it is not, History and Evolution of AI, Current trends in AI, AI and Technology convergence, Industry 4.0 and Digitalization, Role of IoT, Big Data and 5G in AI, Domains of AI – Natural Language Processing, Computer Vision and Statistical Data.

Module 2:

AI Project Cycle-I – Problem Scoping and Data Acquisition, AI Project Cycle-II – Modelling and Evaluation, Societal Impact of AI – ethical concerns and responsible practices, Elements of ML – Supervised Learning, Unsupervised Learning and Reinforcement Learning with examples.

Module 3:

Elements of AI – types and characteristics, Data as Fuel for AI – structured and unstructured data, methods of data mining and data storage, AI/Data Science Team – roles and responsibilities of key stakeholders.

Module 4:

Tools to implement AI – No-Code tools (Teachable Machine, Orange, Roboflow, Chatfuel) and Code-based tools, Introduction to Neural Networks – working of simple Neural Networks, difference between common Deep Learning models with examples and applications.

Module 5:

AI Project-I – Natural Language Processing use case using no-code tool Chatfuel, AI Project-II – Statistical Data use case using no-code tool Orange Data Mining, AI Project-III – Computer Vision use case using no-code tool Roboflow, Future Possibilities of AI – emerging software and hardware technology trends and their direct impact on development of AI.

Labs / Practicals:

1. Familiarize with AI tools and techniques, including statistical data analysis using raw graphs, computer vision applications like Vision API, and generative AI tools such as DALL·E, Stable Diffusion, Gemini, and ChatGPT.
2. Apply the AI Project Cycle to real-world scenarios by identifying problems, collecting data, developing models, and evaluating their effectiveness to propose impactful AI-based solutions.
3. Deconstruct the working behind simple Neural Networks and outline the difference between some common DL models with the help of examples and applications.
4. Explore and utilize no-code tools, such as Teachable Machine, to build AI projects.
5. Project Building using No-code tool – Orange Data Mining for Statistical Data Use cases.
6. Project Building using No-code tool – Roboflow for Computer Vision Use cases.
7. Project Building using No-code tool – Chatfuel for Natural Language Processing Use cases.
8. Classify the type of data into structured and unstructured and evaluate data quality using real datasets.

9. Explore various AI domains by running Vision API experiments and comparing outcomes from NLP, CV, and Statistical Data use cases.
10. Demonstrate the AI Project Cycle on a chosen real-world problem from an industrial or social impact perspective and present findings.

References:

1. Intel AI for Future Workforce – Introduction to AI Course Content, Intel Digital Readiness Program.
2. Stuart Russell and Peter Norvig – Artificial Intelligence: A Modern Approach, Pearson Education.
3. Andreas Mueller and Sarah Guido – Introduction to Machine Learning with Python, O'Reilly Media.
4. Ian Goodfellow, Yoshua Bengio, and Aaron Courville – Deep Learning, MIT Press.
5. Aurélien Géron – Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media.

Course Code	Course Name	L	T	P	Cr
DOAI260003	AI (Industry)-Introduction to Generative AI	3	0	2	4

Course Outcomes:

CO1. Define Gen AI and distinguish its role in the broader context of AI. Describe how the basic model of Gen AI works. Discuss the wider societal impact of Gen AI.
CO2. Understand how NLP has evolved over time and the shift from Traditional AI to Gen AI. Understand Prompt Engineering and its impact on the output of NLP tasks.
CO3. Learn the limitations of current AI generation tools for video and audio. Recognize the challenges in ethical considerations and responsibilities when using generative AI tools.
CO4. Describe the stages of a generative AI project and learn to select and use appropriate AI tools for development.
CO5. Understand the basic structure of neural networks, architecture of transformer models, workings of diffusion models, and the pre-training process for generative AI models.
CO6. Learn to fine-tune generative AI models, understand RLHF, prototype Gen AI applications, and apply principles of responsible and ethical use of LLMs.

Module 1:

Introduction to Gen AI – definition and scope, role of Gen AI in the broader context of AI, how the basic model of Gen AI works, societal impact of Gen AI, Demystifying Gen AI Models – types of generative models (GANs, VAEs, Diffusion Models, LLMs), evolution of NLP to Gen AI, shift from Traditional AI to Gen AI.

Module 2:

Prompt Engineering-I – introduction to prompt engineering, basic applications and impact on NLP task outputs, text generation and conversation initiation, Prompt Engineering-II – advanced prompt engineering techniques for diverse NLP applications, real-world examples such as Mint Mobile Commercial, Basics of Video and Audio for Generation using Gen AI for Productivity-I – introduction to video generation tools (Runway ML, Genmo), text-to-video and image-to-video generation, limitations of current AI generation tools for video and audio.

Module 3:

Basics of Video and Audio for Generation using Gen AI for Productivity-II – audio and music generation using AIVA, code generation using Codeium, text generation using ChatGPT, Gemini and Perplexity, Generative AI-Ethics – ethical considerations, responsibilities and challenges when using generative AI tools, Generative AI Project Lifecycle – stages of a Gen AI project, selection of appropriate AI tools, Gen AI Tools for Development – open-source tools such as Stable Diffusion XL, proprietary tools such as Bing Image Creator powered by DALLE3.

Module 4:

Neural Network Fundamentals – basic structure and functioning of neural networks, how neural networks process data to make predictions, Transformers and the Evaluation of LLMs – architecture of transformer models, methods to evaluate large language models for performance and reliability, walkthrough of OpenAI Playground and parameter exploration, Diffusion Models and the Evaluation of Text to Image Generation – workings of diffusion models, evaluation of quality of text-to-image generation, Pre-Training Gen AI Models-I – pre-training process for generative AI models, importance of pre-training in building effective AI systems, modifying classification labels of Cohere datasets and its impact on fine-tuning and model confidence.

Module 5:

Fine-Tuning Gen AI Models – techniques for fine-tuning generative AI models to adapt them for specific tasks and applications, RLHF – Evaluating and Optimizing Gen AI Models: reinforcement learning with human feedback, evaluation and improvement of performance of generative AI models, Prototyping Gen AI Applications – skills to design, develop and test prototypes of generative AI applications, LangChain components – LLM, Agents, Prompts, Chains, Document loaders and utils with code-based implementation, LLM Application and Responsible use of AI – practical applications of large language models, ethical and responsible use of LLMs, building an AI

Tutor using open-source models such as Dolly 3B and Intel Neural Chat 7B.

Labs / Practicals:

1. Apply prompt engineering skills in basic NLP tasks, such as text generation, conversation initiation, and simple problem-solving.
2. Develop advanced prompt engineering skills for diverse NLP applications. Examine real-world prompt engineering examples, such as the Mint Mobile Commercial, to discern its role in marketing.
3. Explore cutting-edge Gen AI tools, their features, and capabilities for content generation including video and audio.
4. Explore open-source models such as Stable Diffusion XL and proprietary tools such as Bing Image Creator (powered by DALLE3) for image generation.
5. Text-to-video generation and image-to-video generation using tools like Runway ML and Genmo. Generate audio and music files using the tool AIVA.
6. Generate code by prompts using the tool Codeium. Generate text using tools like ChatGPT, Gemini, and Perplexity.
7. Walkthrough of OpenAI Playground. Explore how changing parameters can affect responses generated by different models.
8. Explore how modifying classification labels of Cohere datasets impacts fine-tuning and model confidence.
9. Explore basic components of LangChain like LLM, Agents, Prompts, Chains, and Document loaders and utils, along with a code-based implementation.
10. Utilize open-source text generation models such as Dolly 3B and Intel Neural Chat 7B for building an AI Tutor application.

References:

1. Intel AI for Future Workforce – Introduction to Generative AI Course Content, Intel Digital Readiness Program.
2. Antony Hagiwara – Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play, O'Reilly Media.
3. Ashish Vaswani et al. – Attention is All You Need, Advances in Neural Information Processing Systems.
4. OpenAI – GPT-4 Technical Report, OpenAI.
5. Harrison Chase – LangChain Documentation – https://docs.langchain.com/

Course Code	Course Name	L	T	P	Cr
DOAI260004	AI (Industry)-Introduction to Machine Learning	3	0	2	4

Course Outcomes:

CO1. Describe Machine Learning and how Deep Learning is a subset of the wider field of Machine Learning.
CO2. Understand the importance of dashboards and discuss various applications of dashboards across different industries.
CO3. Explain the mathematical working behind different ML models. Implement different classification and regression ML models using Python.
CO4. Outline the difference between supervised, unsupervised and reinforcement learning algorithms. Examine the working of different reinforcement learning models.
CO5. Describe the working of Neural Networks and contrast it with biological Neurons. Describe various applications of neural networks. Outline different methods to overcome variance and bias in DL models.
CO6. Discuss and interpret the future of ML based on current and upcoming trends. Develop Python-based AI Projects incorporating Supervised Learning, Unsupervised Learning and Deep Neural Networks.

Module 1:

Introduction to Machine Learning – definition, types and applications, Relationship between AI, ML and Deep Learning, Tools to Implement AI (Python & Mathematics) – I: Python basics for ML, NumPy, Pandas, data manipulation.

Module 2:

Tools to Implement AI (Python & Mathematics) – II: mathematical foundations – linear algebra, statistics and probability, No-Code tool for Data Exploration – Tableau dashboard creation, data visualization techniques, statistical concepts implementation using no-code visualization tools.

Module 3:

AI (Modeling) Supervised Learning-I – Linear Regression, Logistic Regression, mathematical working behind models, classification and regression using Python, AI (Modeling) Supervised Learning-II – SVM, Decision Tree, industrial applications of ML models, AI (Modeling) Unsupervised Learning-I – K-Means Clustering, mathematics of clustering algorithms, AI (Modeling) Unsupervised Learning-II – Recommendation Systems, mathematics behind recommendation system, applications of recommendation systems.

Module 4:

Reinforcement Learning – definition, difference from supervised and unsupervised learning, working of reinforcement learning models with applications, Neural Network and Deep Learning-I – biological vs artificial Neurons, working of Neural Networks, applications of neural networks, Deep Learning-II – Convolutional Neural Networks, Recurrent Neural Networks, methods to overcome variance and bias in DL models, implementation using Python and deep learning frameworks, exploring OpenAI's Gym for reinforcement learning.

Module 5:

ML Project – Supervised Learning AI models: end-to-end Python project using classification/regression, ML Project – Unsupervised Learning AI models: end-to-end Python project using clustering, ML Project – Deep Neural Networks: building and training DNN in Python, Future of Machine Learning and Deep Learning – current and upcoming trends in ML and DL.

Labs / Practicals:

1. Interpret and understand the basic tools required for building ML Projects through a selected list of topics from Mathematics and Python Programming.
2. Develop and create a simple dashboard for visualizing data through Tableau.
3. Implement statistical concepts using the no-code visualization tool and create your own dashboard with the help of a no-code visualization tool.

4. Implement supervised machine learning algorithms, such as SVM and Decision Tree, using Python to develop a deeper understanding of their practical applications.
5. Implement unsupervised machine learning algorithms, such as K-Means, using Python to develop a deeper understanding of their practical applications.
6. Implement neural networks and create simple generative AI models using Python and deep learning frameworks, including exploring OpenAI's Gym for hands-on experience with AI and reinforcement learning.
7. Develop Python-based use cases and AI Projects incorporating Supervised Learning methods.
8. Develop Python-based use cases and AI Projects incorporating Unsupervised Learning methods.
9. Develop Python-based use cases and AI Projects incorporating Deep Neural Networks.
10. Implement a recommendation system using Python and evaluate its performance on a real-world dataset.

References:

1. Intel AI for Future Workforce – Introduction to Machine Learning Course Content, Intel Digital Readiness Program.
2. Andreas Mueller and Sarah Guido – Introduction to Machine Learning with Python, O'Reilly Media.
3. Ian Goodfellow, Yoshua Bengio, and Aaron Courville – Deep Learning, MIT Press.
4. Aurélien Géron – Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media.
5. Christopher Bishop – Pattern Recognition and Machine Learning, Springer.

Course Code	Course Name	L	T	P	Cr
DCSA250057	Object Oriented Programming using C++	3	0	2	4

Course Outcomes:

CO1. Describe the different features of Object Oriented Programming.
CO2. Write programs to Implement the concepts of classes and objects.
CO3. Create new classes using the concepts of inheritance.
CO4. Apply knowledge of Polymorphism to solve real life problems.
CO5. Implement exception handling mechanism.

Module 1:

Introduction to Object Oriented Programming: Object Oriented Paradigm Objects and Classes, Features Object oriented Programming, Structured Vs Object Oriented Development, Features of Object Oriented Languages, Applications ofObjectOriented Programming, Merits and Limitations of Object Oriented Programming.Basic Data types, Basic Type modifiers, Derived Data types, Variables, Storage class specifiers, Initializing variables, Operators, Unformatted Console and stream I/O Functions, Formatted Console I/O Functions

Module 2:

Classes and Objects: Classes ,Class Members and Creating Objects, Member functions, Member Access Specifiers (public, private, protected), Static class member, Inline Functions, Arrays within a Class and Array of Objects, Passing Objects as function arguments and returning object from a function,Constructors, Overloaded Constructors, Null Contradictor, Copy Constructor, Destructors Constraints on Constructors and Destructors

Module 3:

Inheritance: Base and Derived classes, Accessing Base class members and Access Control, Overriding member functions, Multi Level, Multiple, Hierarchical& Hybrid Inheritance, Virtual Base Class.

Module 4:

Polymorphism: Fundamental of Polymorphism, Overloading Functions, Overloading Operators (Unary, binary, string manipulation using operator), Pointer to object and derived class, 'This' pointer, Virtual Functions, Early and Late Binding, Rules of Virtual Functions, Pure Virtual Function, Friend Functions.

Module 5:

File Handling, Exception Handling & Templates: Basic File Operations, File Handling, Classes for file stream operation, Opening and Closing Files, File modes File, Introduction to Exception Handling, Catching Class Types, Multiple Catch Handlers, Exception Specification, Generic Functions/Function Templates, Template Arguments.

Labs / Practicals:

1. Write a C++ program to demonstrate class and object creation.
2. Implement constructors and destructors in a C++ class.
3. Demonstrate function overloading and operator overloading in C++.
4. Create a program to implement single and multilevel inheritance.
5. Use virtual functions to implement runtime polymorphism.
6. Design a template class for stack or queue operations.
7. Demonstrate file handling using file streams in C++.
8. Implement a program using exception handling with multiple catch blocks.

References:

1. Object Oriented Programming With C++, E Balagurusamy
2. Object oriented programming in C++ , Robert Laffore
3. Introduction To Programming With C++, Diane Zak
4. Object oriented programming with C++ Reema Thareja

Course Code	Course Name	L	T	P	Cr
DCSA250059	Object-Oriented Programming with Java	2	0	4	4

Course Outcomes:

CO1: Describe the procedural and object oriented paradigm with concepts of streams, classes, functions, data and objects.
CO2: Understand dynamic memory management techniques using pointers, constructors, destructors, etc
CO3: Describe the concept of function overloading, operator overloading, virtual functions and polymorphism.
CO4: Classify inheritance with the understanding of early and late binding, usage of exception handling, generic programming.
CO5: Demonstrate the use of various OOPs concepts with the help of program

Module 1:

An Overview of Java
Object-Oriented Programming, A First Simple Program, A Second Short Program, Two Control Statements, Using Blocks of Code, Lexical Issues, The Java Class Libraries, Data Types, Variables, and Arrays: Java Is a Strongly Typed Language, The Primitive Types, Integers, Floating-Point Types, Characters, Booleans, A Closer Look at Literals, Variables, Type Conversion and Casting, Automatic Type Promotion in Expressions, Arrays, A Few Words About Strings, Arithmetic Operators, The Bitwise Operators, Relational Operators, Boolean Logical Operators, The Assignment Operator, The ? Operator, Operator Precedence, Using Parentheses, Control Statements: Java's Selection Statements, Iteration Statements, Jump Statements.

Module 2:

Object and Class
Class Fundamentals, Declaring Objects, Assigning Object Reference Variables, Introducing Methods, Constructors, The this Keyword, Garbage Collection, The finalize( ) Method, A Stack Class, A Closer Look at Methods and Classes: Overloading Methods, Using Objects as Parameters, A Closer Look at Argument Passing, Returning Objects, Recursion, Introducing Access Control, Understanding static, Introducing final, Arrays Revisited, Inheritance: Inheritance, Using super, Creating a Multilevel Hierarchy, When Constructors Are Called, Method Overriding, Dynamic Method Dispatch, Using Abstract Classes, Using final with Inheritance, The Object Class.

Module 3:

Interface and Packages
Defining Interfaces, Extending Interfaces, Implementing Interfaces, Accessing Interface Variables. Java API Packages, Using System Packages, Naming Conventions, Creating Packages, Accessing a Package, Using a Package, Adding a Class to a Package, Hiding Classes, Static Import.

Module 4:

Exception Handling and I/O Exceptions
Handling-Fundamentals, exception types, using try and catch, multiple catch clause, nested try statements – throw, throws and finally, built-in exceptions, creating own exceptions. Input / Output Basics – Streams – Byte streams and Character streams – Reading and Writing Console – Reading and Writing Files

Module 5:

Multithreaded Programming
Creating Threads, Extending the Thread Class, Stopping and Blocking a Thread, Life Cycle of a Thread, Using Thread Methods, Thread Exceptions, Thread Priority, Synchronization, Implementing the 'Runnable' Interface,

Inter-thread communication. vectors, lists, maps.

Labs / Practicals:

Preparing

References:

1.Java: The Complete Reference, Twelfth Edition Paperback – 23 December 2021 by Herbert Schildt
2.Java Performance: The Definitive Guide: Getting the Most Out of Your Code by Scott Oaks

Course Code	Course Name	L	T	P	Cr
DCSE250113	Programming in Python	2	0	4	4

Course Outcomes:

CO1: Remember syntax rules, basic operators, and built-in features of Python 3.

CO2: Understand guiding principles and coding conventions from Zen of Python.

CO3: Apply object-oriented programming paradigms in design and practice.

CO4: Analyze the runtime efficiency and resource utilization of Python programs.

CO5: Evaluate the outcomes of programs manually with dry runs and trace tables.

CO6: Create modular and maintainable programs following Pythonic practices.

Module 1:

Constants, types, identifiers, keywords, comments, print, input, help functions.
Operator precedence and associativity, arithmetic operators (**, *, /, //, %, +, -).
Binary number system, bin function, bitwise operators (shift, &, ^, |), bit masking.
Relational and equality operators (== vs "is"), not, short-circuit evaluation (and, or).

Module 2:

Selection: if, elif, else, mutual exclusion, indent (space vs tab), nesting, pass.
Iteration: while loop, range of integers, in operator, for loop, break vs continue.
Function: parameters, arguments, return value, global, recursion, lambda, closure.
Modules: from, import, as, __name__, top-level module, avoiding circular imports.

Module 3:

ASCII and UTF-8, ord and chr functions, escape sequence, strings, docstring, regex.
Mutability, id function, object vs reference, is vs ==, interning, datetime module.
Ordered: list, tuple, negative indexing, slicing, sort with key, pack and unpack.
Unordered: set, dictionary (keys, values, items), subscript vs get method, None.

Module 4:

Class definition, self parameter in methods, constructor, attributes, instantiation.
Dot notation, type function, __str__ vs __repr__, methods for built-in operators.
Static attributes, annotations, private name mangling, getter and setter methods.
Inheritance, object, super, C3 linearization, ambiguous MRO, abstract base class.

Module 5:

Syntax errors vs runtime exceptions, traceback, try, except, else, finally, raise.
Matching except clause, except with multiple exceptions, finally with return.
Built-in exceptions, user-defined exception classes, exceptions with arguments, as.
Command-line arguments, file opening modes (read/write/append), with statement.

Labs / Practicals:

01. Test the precedence and associativity of arithmetic operators.

02. Test the bitwise operators and verify the results with bin function.

03. Test the relational operators (with and without chaining).

04. Test the short-circuit evaluation of and, or operators using function calls as operands.

05. Convert from metric prefix (KB/MB/GB/TB) to binary prefix (KiB/MiB/GiB/TiB).

06. Use if-elif-else ladder to print the grade as per the score obtained in a subject.

07. Find the mean, median, mode, variance, and standard deviation for a list of scores.

08. Print values using additive/subtractive rules of roman numerals: I, V, X, L, C, D, M.

09. Use string repetition with multiplication to generate patterns without nested loops.

10. Use nested loops to generate asymmetric and symmetric patterns of arbitrary size.

11. Check whether a given string is a palindrome (or not) without reversing the string.

12. Test positional arguments, default arguments, and keyword arguments for functions.

13. Write a function to check whether an indexed collection of numbers is already sorted.

14. Write a function for primality testing, and use it to find first n pairs of twin primes.

15. Write a function to generate the Collatz sequence (up to 1) for a positive integer.

16. Generate the terms of Fibonacci sequence using iterative and recursive approaches.

17. Print optimal sequence of moves for a given instance of “Towers of Hanoi” puzzle.

18. Find all combinations to make a given payment from a limited set of coin types.

19. Check if a letter is a vowel using two approaches: if-elif-else and in operator.

20. Find vowel count in a string using a lookup table for case-insensitive vowel check.

21. Write a regular expression to check whether a string is a valid email ID (or not).

22. Use case-insensitive and dot-insensitive matching to check if two email IDs are same.

23. Display a countdown timer in HH:MM:SS format using carriage return ("\r").

24. Get current time and find clockwise angles of hour hand, minute hand, second hand.

25. Find the day of week for any date in Gregorian calendar using doomsday algorithm.

26. Define Matrix class and implement these operators: unary +, unary -, +, -, *, ==.

27. Define Person class in a module and extend it with Student class in another module.

28. Incrementally generate roll numbers in Student constructor using a static counter.

29. Populate a list with Student instances and write their attributes in a new CSV file.

References:

1. "Python Tutorial" by Guido van Rossum and the Python development team
https://bugs.python.org/file47781/Tutorial_EDIT.pdf

2. "Think Python: How to Think Like a Computer Scientist (Second Edition)" by Allen Benjamin Downey
https://greenteapress.com/thinkpython2/thinkpython2.pdf

3. "Python for Everybody: Exploring Data in Python 3" by Charles Russell Severance
https://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf

4. "Beej's Guide to Python Programming" by Brian Hall
https://beej.us/guide/bgpython/pdf/bgpython_a4_c_2.pdf

5. "Learning Python: Powerful Object-Oriented Programming (Sixth Edition)" by Mark Lutz
https://learning-python.com

6. "Fluent Python: Clear, Concise, and Effective Programming (Second Edition)" by Luciano Ramalho
https://www.fluentpython.com

Course Code	Course Name	L	T	P	Cr
DOEE250078	Electronic Devices and Circuits	3	0	2	4

Course Outcomes:

CO1: Understand the fundamentals of semiconductors, energy bands, carrier concentrations, and analyze P-N junction diode behavior including Zener diodes, LEDs, and photodiodes.
CO2: Apply diode principles to design and analyze wave-shaping circuits including RC differentiators, integrators, clippers, clampers, and voltage multipliers.
CO3: Design and evaluate rectifier circuits (half-wave, full-wave, bridge) with appropriate filters, and analyze their performance parameters such as ripple factor, efficiency, and load regulation.
CO4: Analyze the operation of Bipolar Junction Transistors (BJT), including CE, CB, and CC configurations, understand biasing techniques, equivalent circuits, and use BJTs for amplification and switching applications.
CO5: Understand and analyze Field Effect Transistors (JFET, MOSFET, FinFET), including device characteristics, non-ideal effects, CV characteristics, and their applications in analog and digital circuits.

Module 1:

Semiconductor Physics and Diode Theory: Introduction to Semiconductors: Intrinsic and Extrinsic semiconductors, Energy bands in solids, Conductors, Semiconductors and Insulators, Carrier concentration in intrinsic and extrinsic semiconductors, P-N junction diode: formation and characteristics, Current components in a diode, diode equation, Zener diode and its applications, Photodiode and LED.

Module 2:

Diode Applications-Wave shaping circuits:RC differentiating and integrating circuits. Analysis of first order RC differentiator and integrator to step input, rise time, bandwidth. Diode Clippers: Series and parallel configurations, biased clippers, Practical clippers and their transfer characteristics, Diode Clampers: Positive and negative clamping, Applications in signal shaping.
Voltage doubler, tripler, and quadrupler circuits, Zener Diode as a Voltage Regulator: Operation, load and line regulation, Series and shunt voltage regulators (brief overview).

Module 3:

Diode Applications- Rectifiers and Filters: Rectifiers and Filters: Half-Wave Rectifier: Operation, waveform analysis, ripple factor, efficiency, Full-Wave Rectifier (Center-tap and Bridge): Analysis, performance comparison, Filter circuits: Capacitor filter, inductor filter, π-filter, Load and transformer utilization factors.

Module 4:

Bipolar junction transistor: Bipolar Junction Transistors: Fundamental operation, Input and output characteristics of BJT in active region, BJT configurations: CE, CB, CC. Amplification with BJTs, generalized biasing and equivalent circuit models, non-ideal effects, switching.

Module 5:

Field Effect Transistor: Field – Effect Transistors: Transistor operations. JFET, MOSFET and their operations, device characteristics, non-ideal effects, CV characteristics, equivalent circuits. Overview of FinFET.

Labs / Practicals:

1. RC integrator and differentiator circuits: To design and implement RC integrator and differentiator circuits using a square wave input of period ‘T’, and to analyze the output waveforms for the following conditions:
- Time constant (τ) much greater than T
- Time constant much less than T
- Time constant approximately equal to T

2. Characterization of Semiconductor Materials: Perform experiments to understand the electrical properties of semiconductor materials such as silicon and germanium. Measure parameters like resistivity, mobility, and carrier concentration.
3. PN Junction Diode Characteristics: Study the I-V characteristics of a PN junction diode under forward and reverse bias conditions. Determine parameters like threshold voltage, forward and reverse bias currents, and ideality factor.
4. Diode Clipping and Clamping Circuits: To design and implement various diode-based wave shaping circuits such as series and parallel clippers (biased and unbiased) and clamping circuits, and to observe their effects on input waveforms.
5. Filter circuits: To design and construct filter circuits (capacitor filter, LC filter, and π-filter)
6. Zener Diode Characteristics: Investigate the voltage-regulating properties of Zener diodes. Measure the breakdown voltage and dynamic resistance of Zener diodes under different load conditions.
7. Zener Diode as a Voltage Regulator
8. Diode Rectifier Circuits: Construct and analyze various diode rectifier circuits such as half-wave, full-wave bridge, and center-tapped full-wave rectifiers. Measure output voltage, ripple factor, and efficiency.
9. Bipolar Junction Transistor (BJT) Characteristics: Study the DC and AC characteristics of NPN and PNP bipolar junction transistors. Measure parameters like DC current gain (β), collector current vs. collector-emitter voltage (IC- VCE) characteristics, and output characteristics.
10. BJT Amplifier Circuits: Design and analyze common-emitter and common-base amplifier circuits using bipolar junction transistors. Measure parameters like voltage gain, input/output impedance, and frequency response.
11. Field Effect Transistor (FET) Characteristics: Investigate the DC and AC characteristics of both JFET and MOSFET transistors. Measure parameters like drain current vs. drain-source voltage (ID-VDS), transconductance, and output conductance.
12. FET Amplifier Circuits: Design and analyze common-source and common-drain amplifier circuits using field-effect transist

References:

1. Electronic devices and Circuit Theory", "R. Boylestad", "Pearson Education", 9th Edition
2. "Electron devices", "S. Poornachandra, Sasikala", "Scitech", 2nd Edition
3. "Electronic Devices and Circuits", "Millman Halkias", "TMH", 2000
4. "Electronic Devices and Circuits", "DavidA.Bell", "PHI", 4thEdition

Course Code	Course Name	L	T	P	Cr
DOAI250005	Artificial Intelligence and Machine Learning	2	0	4	4

Course Outcomes:

CO1: Understand AI problem-solving techniques, heuristic search algorithms, and reinforcement learning.
CO2: Apply knowledge representation, logic-based reasoning, and probabilistic models in AI systems.
CO3: Develop and evaluate machine learning models using supervised and unsupervised learning techniques.
CO4: Design and implement deep learning architectures for vision, language, and sequential data applications.
CO5: Analyze advanced AI methods, ethical challenges, and emerging trends in artificial intelligence.

Module 1:

AI Techniques and Search Strategies
History of AI, problem-solving using search, uninformed vs. informed search, heuristic search techniques (Hill Climbing, Simulated Annealing, A* Algorithm), adversarial search (Minimax Algorithm, Alpha-Beta Pruning), AI-driven game strategies (Chess, Chinese Checkers), Reinforcement Learning fundamentals (Markov Decision Processes, Q-learning), heuristic optimization techniques (Genetic Algorithms, Particle Swarm Optimization).

Module 2:

Knowledge Representation, Reasoning, and Probabilistic Models
Propositional logic, inference mechanisms, forward and backward chaining, probabilistic reasoning techniques (Bayesian Networks, Markov Models, Probabilistic Graphical Models), logic-based AI vs. machine learning-driven reasoning, Natural Language Processing (NLP) fundamentals (tokenization, embeddings, language models), AI-driven text processing, machine translation.

Module 3:

Machine Learning Fundamentals and Neural Networks
Supervised and unsupervised learning paradigms, classification and regression techniques (decision trees, support vector machines, ensemble learning methods: Random Forest, XGBoost), artificial neural networks (activation functions, loss functions, back propagation, multilayer perceptions), overfitting, bias-variance tradeoff, hyperparameter tuning, performance evaluation metrics (precision-recall, ROC curves, F1-score).

Module 4:

Deep Learning and Advanced Neural Architectures
Convolutional Neural Networks (CNNs) for image classification and object detection (convolution layers, pooling, fully connected layers), Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks for sequential data processing, Transformer models (BERT, GPT) for NLP applications, optimization techniques (batch normalization, dropout regularization, transfer learning), deep learning applications (medical imaging, self-driving vehicles, generative AI).

Module 5:

Unsupervised Learning, Generative Models, and Ethical AI
Clustering techniques (K-Means, Hierarchical Clustering), Autoencoders, Generative Adversarial Networks (GANs), self-supervised learning, foundation models in AI, AI applications in healthcare, finance, and autonomous systems, AI ethics (fairness, bias mitigation, adversarial attacks, explainable AI), advancements in AI (quantum computing, federated learning, AI governance frameworks).

Labs / Practicals:

1. Implement a basic uninformed search algorithm (e.g., Breadth-First Search, Depth-First Search).
2. Implement informed search algorithms using heuristics (e.g., A* Algorithm for pathfinding).
3. Implement the Hill Climbing algorithm for local search problems.
4. Develop a solution using Simulated Annealing for optimization problems.
5. Implement the Minimax algorithm to solve a simple game (e.g., Tic-Tac-Toe).
6. Implement Alpha-Beta Pruning for optimizing the Minimax algorithm.
7. Build a Reinforcement Learning agent using Q-learning to solve a basic maze problem.
8. Implement Genetic Algorithms to solve optimization problems (e.g., travelling salesman).
9. Build a Particle Swarm Optimization model for function optimization.
10. Implement propositional logic and inference mechanisms (e.g., forward and backward chaining).
11. Develop a basic Bayesian Network for probabilistic reasoning.

References:

1.Tom M. Mitchell: Machine Learning. McGraw Hill
2.Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig
3.Pattern Recognition and Machine Learning by Christopher M. Bishop
4.Machine Learning: A Probabilistic Perspective by Kevin P. Murphy
5.Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto
6.Natural Language Processing with Python by Steven Bird, Ewan Klein, and Edward Loper
7.The Elements of Statistical Learning" by Trevor Hastie, Robert Tibshirani, and Jerome Friedman
8.Michael Nielsen: Neural Networks and Deep Learning (free online book)
9.Stuart Russel & Peter Norvig: Artificial Intelligence– A Modern Approach. Pearson, 4th edition, 2023
10.Deep Learning (Adaptive Computation and Machine Learning series) Illustrated Edition, by Ian Goodfellow, Yoshua Bengio, Aaron Courville, MIT Press, London

Course Code	Course Name	L	T	P	Cr
DOAI250021	Data Science	3	0	2	4

Course Outcomes:

CO1: Describe the data science life cycle, key concepts, and the role of Python in data acquisition, preprocessing, and visualization.
CO2: Apply core Python programming constructs and data structures (lists, dictionaries, tuples, stacks, queues, data frames) for effective data manipulation and analysis.
CO3: Develop and utilize user-defined functions and classes to implement modular and object-oriented programming techniques in data science workflows.
CO4: Analyze and handle data using file operations, pandas data frames, and essential machine learning libraries such as NumPy, Scikit-learn, and Matplotlib.
CO5: Implement and evaluate machine learning models using Scikit-learn for classification, regression, clustering, and present insights through case studies like spam detection and customer segmentation.

Module 1:

Data Science Overview: Data Science life cycle; data acquisition, public data repositories, pre-processing, data visualization, Machine Learning algorithms, pattern evaluation, knowledge representation, analytical strategies; Python fundamentals: data types, mutable & immutable types, operators & expressions, standard library modules, the flow of control, etc.

Module 2:

Python data structures: String manipulation: string operators, slices, methods; List manipulation: creating & accessing the list, list operations, true copy of lists, nested lists; Tuples: creation, accessing, modifying tuples, nested tuples, methods & functions, tuple slicing; Dictionaries: key-value pairs, creation, accessing, modifying dictionaries, Stack, Queue, data frame, etc.

Module 3:

User-defined functions & Classes: modules in Python, importing python modules, name resolution, module aliasing, package/library, locating modules, using Python's standard library, user-defined functions: parameters in functions, passing arrays/tuples/dictionary, and lists to functions, function call, the scope of variables, dynamic typing and dynamic binding, Creating and instantiating classes with attributes and methods, public v/s non-public members, Inheritance, Polymorphism, etc.

Module 4:

Data Handling: data file operations: opening & closing a file, reading/writing/appending a file, relative and absolute paths, standard file streams, binary file operations, CSV file handling & pickling, concatenating multiple CSV files/text files, panada's DataFrame, convert pandas data frames to numpy arrays/CSV, etc; introduction to various ML libraries: NumPy, Pandas, Scikit-learn, SciPy, Statsmodels, MatplotLib, Seaborn, Sympy, etc. and their use cases.

Module 5:

Scikit-learn library: using sci-kit-learn modules for classification e.g. Support Vector Classifier & Regressor, CART, clustering (k-means), data reduction (PCA), data visualization (box plot), etc. Web Scraping with Beautiful Soup, use cases and case studies: spam detection, drug response, stock price prediction, customer segmentation, etc.

Labs / Practicals:

1. Drawing different patterns e.g. Pascal triangle, Pyramid of '*', etc. using loop and branching statements
2. Creating a Python class, instantiate 10 objects, save them in csv file, re-open and load them in Python workspace
3. Sorting a student database based on student identification numbers by reading it from csv file

4. Problem related to classification using CART, SVC, K-NN, etc
5. Problem e.g. Market-Basket analysis, related to association mining using Apriori, FPGrowth, etc.
6. Problem related to clustering e.g. image segmentation using K-means,
7. Problem related to regression analysis e.g. electricity load prediction during peak hours using SVR, CART, etc.
8. Problems related to anomaly detection e.g.
9. Training a Machine learning model for a real-life dataset with good practices such as Train/Test split, 10-fold cross validation, reporting performance with RoC plots etc. over a separate test dataset
10. Using all plot libraries to do data pre-processing & visualization e.g. Box plot, Bar chart, probability density function plot, etc.
11. Understanding data processing, reading, pre-processing, cleaning data set, generating Train/Test/Validation datasets, label encoding, one-hot encoding, etc. using Sklearn
12. Importing & using various ML libraries: NumPy, Pandas, Scikit-learn, SciPy, Statsmodels, MatplotLib, Seaborn, Sympy, etc.
13. Creating and using NumPy Array and converting it into other types
14. Creation and usage of panda's data frame and converting it into other types
15. Using various data visualization techniques such as probability density function, box plot, etc.
16. CSV file handling & pickling, concatenating multiple CSV files/text files
17. Implementing a regression algorithm for rain-fall prediction, stock market prediction, wine quality, etc.
18. Implementing a classification algorithm for image classification, breast cancer, etc.
19. Implementing an unsupervised algorithm for grouping data points into meaningful clusters.
20. Implementing a clustering algorithm for data reduction

References:

1. Cormen, Leiserson, Rivest and Stein, Introduction to algorithms (Main textbook)
2. Kleinberg and Tardos , Algorithm Design
3. Mark Weiss, Data structures and algorithm analysis in C++ (Java)
4. Aho, Hopcroft and Ullman, Data structures and algorithms
5. S. Sahni, Data Structures, Algorithms, and Applications in C++, Silicon Press

Course Code	Course Name	L	T	P	Cr
DCSE250040	Concurrent and Parallel Processing	3	0	2	4

Course Outcomes:

CO1. Understand the concepts of concurrency, parallelism, and their relevance to modern computing.
CO2. Analyze and implement synchronization mechanisms to manage shared resources.
CO3. Design and evaluate multithreaded and parallel programs using programming models.
CO4. Apply parallel computing architectures, patterns, and performance tuning strategies.
CO5. Utilize parallel and concurrent programming tools for solving real-world computational problems.

Module 1:

Introduction to Concurrency and Parallelism:
Concepts of concurrent and parallel execution, Process and thread models, Types of parallelism – data and task parallelism, Challenges in parallel programming, Granularity, Performance metrics – speedup, scalability, Amdahl's law, Gustafson's law.

Module 2:

Synchronization and Coordination Techniques:
Critical section problem, Mutual exclusion – locks, semaphores, monitors, Condition variables, Classical problems – producer-consumer, readers-writers, dining philosophers, Deadlock – detection, prevention, avoidance, Inter-process communication – message passing, shared memory.

Module 3:

Parallel Programming Models and Frameworks:
Shared memory programming – Pthreads, OpenMP; Distributed memory programming – MPI; GPU programming basics – CUDA/OpenCL; Programming constructs – threads, parallel loops, synchronization, data dependencies, reduction, barriers.

Module 4:

Parallel System Architectures and Scheduling:
Parallel computing models – SIMD, MIMD, SMP, multicore, manycore, GPU, cluster; Flynn's taxonomy; Thread/process scheduling – static and dynamic, load balancing, scheduling granularity; Cache coherence, memory consistency models.

Module 5:

Performance Evaluation and Applications:
Performance analysis – execution time, speedup, efficiency, overhead, throughput, latency; Profiling and debugging tools – GProf, Valgrind, Perf, Intel VTune; Applications – scientific computing, real-time systems, machine learning workloads, big data processing.

Labs / Practicals:

1. Implement producer-consumer problem using semaphores in C/Python.
2. Solve the dining philosophers problem using threads and mutexes.
3. Develop a multithreaded application using Pthreads (e.g., file search or prime number finder).
4. Write OpenMP programs for parallel matrix multiplication and vector addition.
5. Use MPI for inter-process communication in a distributed system (e.g., sum of array elements).
6. Compare the performance of sequential and parallel versions of quicksort or mergesort.
7. Implement deadlock detection and avoidance in a simulated resource allocation system.
8. Profile a multithreaded program using GProf or Valgrind and analyze bottlenecks.
9. Use CUDA or OpenCL to perform basic GPU parallel computation (e.g., vector addition).
10. Mini-project: Real-time concurrent log monitoring system using threads and shared memory.

References:

1. Peter Pacheco – An Introduction to Parallel Programming, Morgan Kaufmann.
2. Barry Wilkinson and Michael Allen – Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, Pearson.
3. Andrew S. Tanenbaum – Modern Operating Systems, Pearson (Concurrency Sections).
4. David R. Butenhof – Programming with POSIX Threads, Addison-Wesley.
5. Michael J. Quinn – Parallel Programming in C with MPI and OpenMP, McGraw-Hill.

Course Code	Course Name	L	T	P	Cr
DCSE250068	Distributed Computing and Networking	3	0	2	4

Course Outcomes:

CO1. Understand the fundamental principles of distributed computing and its architecture.
CO2. Analyze inter-process communication, remote procedure calls, and distributed synchronization mechanisms.
CO3. Examine consistency models, fault tolerance, and replication strategies in distributed systems.
CO4. Explore distributed file systems and middleware solutions used in modern distributed applications.
CO5. Evaluate real-world distributed applications and implement basic distributed algorithms and network protocols.

Module 1:

Introduction to Distributed Systems:
Definition and characteristics of distributed systems, Goals – transparency, openness, scalability, fault tolerance; System models – architectural, interaction, and failure models; Client-server and peer-to-peer models; Examples of distributed systems – DNS, cloud platforms.

Module 2:

Communication and Coordination in Distributed Systems:
Inter-process communication – message passing, sockets, RPC, RMI; Group communication and multicast; Time and global state – physical clocks, logical clocks (Lamport, vector clocks), clock synchronization – Cristian's and Berkeley's algorithms; Coordination and agreement.

Module 3:

Synchronization, Consistency, and Replication:
Mutual exclusion – centralized, distributed, token-based algorithms; Election algorithms – Bully, Ring; Deadlock detection in distributed systems; Consistency models – strict, sequential, causal; Replication – passive and active, quorum-based protocols.

Module 4:

Distributed File Systems and Middleware:
Distributed file system architecture – NFS, AFS, HDFS, GFS; Naming services; Directory and file access protocols; Middleware – CORBA, Java RMI, Message Oriented Middleware (MOM), load balancing in distributed systems.

Module 5:

Advanced Topics in Distributed Computing and Networking:
Cloud and edge computing; Distributed databases – CAP theorem, consistency models (ACID vs BASE); Software-defined networking (SDN); Overlay networks and peer-to-peer systems; Case studies – Apache Kafka, Hadoop, BitTorrent; Security and trust in distributed environments.

Labs / Practicals:

1. Implementation of client-server communication using TCP sockets in Python/Java.
2. Demonstration of Remote Procedure Call (RPC) mechanism using gRPC or Java RMI.
3. Simulation of Lamport's logical clock for event ordering in a distributed system.
4. Implementation of the Bully election algorithm for leader election.
5. Design of a distributed mutual exclusion mechanism using Ricart-Agrawala algorithm.
6. Time synchronization simulation using Berkeley or Cristian's algorithm.
7. Experiment on distributed file sharing using Hadoop HDFS (basic file I/O operations).
8. Developing a simple publish-subscribe messaging system using Apache Kafka.

9. Simulation of quorum-based replication with consistency checks.
10. Mini-project: Implementing a distributed chat system with peer discovery and fault tolerance.

References:

1. Andrew S. Tanenbaum and Maarten van Steen – Distributed Systems: Principles and Paradigms, Pearson Education.
2. George Coulouris, Jean Dollimore, Tim Kindberg – Distributed Systems: Concepts and Design, Pearson Education.
3. M. L. Liu – Distributed Computing: Principles and Applications, Pearson.
4. Pradeep K. Sinha – Distributed Operating Systems: Concepts and Design, PHI.
5. Ajay D. Kshemkalyani and Mukesh Singhal – Distributed Computing: Principles, Algorithms, and Systems, Cambridge University Press.

Course Code	Course Name	L	T	P	Cr
DCSA250038	Introduction to Blockchain Technology	2	1	2	4

Course Outcomes:

CO1. Understand the foundational concepts of blockchain technology, including distributed systems and cryptographic principles.
CO2. Analyze and differentiate between various consensus mechanisms and their use in blockchain networks.
CO3. Explore and implement smart contracts using platforms like Ethereum.
CO4. Evaluate blockchain applications in domains such as finance, supply chain, and digital identity.
CO5. Develop, deploy, and test basic blockchain-based decentralized applications (DApps).

Module 1:

Fundamentals of Blockchain Technology:
Introduction to blockchain, History and evolution, Distributed ledger technology, Peer-to-peer networks, Blocks and chains, Transactions and structure of a block, Hashing, Mining, Merkle trees, Blockchain types – public, private, consortium.

Module 2:

Cryptography and Consensus Mechanisms:
Hash functions – SHA-256, Digital signatures, Public and private key cryptography, Elliptic curve cryptography, Consensus algorithms – Proof of Work (PoW), Proof of Stake (PoS), Delegated PoS, Practical Byzantine Fault Tolerance (PBFT), Security and attack models.

Module 3:

Bitcoin and Ethereum Architecture:
Bitcoin architecture – transactions, UTXO model, mining, block structure, wallets; Ethereum architecture – account model, Ethereum Virtual Machine (EVM), gas concept, Ethereum transactions, comparison between Bitcoin and Ethereum, wallet types and key management.

Module 4:

Smart Contracts and Decentralized Applications (DApps):
Introduction to smart contracts, Solidity programming basics, Writing and deploying smart contracts, Contract lifecycle, Interacting with contracts via Web3.js, Truffle framework, MetaMask, Design of DApps, Frontend-backend integration for blockchain applications.

Module 5:

Blockchain Applications and Future Trends:
Blockchain applications in supply chain, healthcare, finance, identity management, voting systems; Blockchain in IoT and AI integration; Introduction to Hyperledger Fabric and Corda; Interoperability and scalability challenges, Layer 2 solutions, Sidechains, Rollups, Regulatory and ethical considerations.

Labs / Practicals:

1. Installation and setup of a private Ethereum blockchain using Ganache.
2. Creating and managing Ethereum wallets using MetaMask.
3. Writing and deploying a basic Solidity smart contract (e.g., "Hello Blockchain").
4. Implementing a voting smart contract and interacting via Web3.js.
5. Developing a token using the ERC-20 standard.
6. Building a decentralized To-Do list DApp using Truffle framework.
7. Simulating a blockchain network with multiple nodes using Ganache CLI.
8. Exploring blockchain explorer tools (e.g., Etherscan) to trace transactions.
9. Creating a multisignature wallet smart contract.

10. Demonstrating a blockchain-based supply chain prototype with product tracking.

References:

1. Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder – Bitcoin and Cryptocurrency Technologies, Princeton University Press.
2. Melanie Swan – Blockchain: Blueprint for a New Economy, O'Reilly Media.
3. Andreas M. Antonopoulos – Mastering Bitcoin: Programming the Open Blockchain, O'Reilly Media.
4. Andreas M. Antonopoulos and Gavin Wood – Mastering Ethereum, O'Reilly Media.
5. Imran Bashir – Mastering Blockchain, Packt Publishing.

Course Code	Course Name	L	T	P	Cr
DCSE250020	Cloud Computing	3	0	2	4

Course Outcomes:

CO1: Explain the evolution, architecture, and key characteristics of cloud computing, including service models (IaaS, PaaS, SaaS) and deployment types (public, private, hybrid).

CO2: Describe various types and implementation levels of virtualization, including CPU, memory, storage, and network virtualization in cloud environments.

CO3: Analyze layered cloud architecture and address design challenges related to inter-cloud resource management and platform deployment.

CO4: Apply distributed and parallel programming paradigms such as MapReduce and Hadoop for cloud-based application development and understand various cloud platforms like AWS, OpenStack, and Google App Engine.

CO5: Evaluate cloud security challenges and solutions, including data and application security, VM security, and risk management strategies in cloud environments.

Module 1:

Introduction to Cloud Computing

Evolution of Cloud Computing, System Models for Distributed and Cloud Computing, NIST Cloud Computing Reference Architecture, Features of Cloud Computing, Cloud Services – IaaS, PaaS, SaaS, Cloud service Providers – Public , Private and Hybrid Clouds.

Module 2:

Introduction to component virtualization

Basics of Virtualization, Types of Virtualization, Implementation Levels of Virtualization, Virtualization of CPU, Memory, I/O Devices, Desktop Virtualization, Server Virtualization, Storage Virtualization, Network Virtualization.

Module 3:

Architectural Design of Compute and Storage Clouds

Layered Cloud Architecture Development, Design Challenges, Inter Cloud Resource Management, Resource Provisioning and Platform Deployment, Global Exchange of Cloud Resources.

Module 4:

Parallel and Distributed Programming Paradigms

Map Reduce, Twister and Iterative MapReduce, Hadoop Library from Apache, Mapping Applications, Programming Support, Google App Engine, Amazon AWS, Cloud Software Environments - Eucalyptus, Open Nebula, OpenStack.

Module 5:

Security Overview

Cloud Security Challenges, Software-as-a-Service Security, Security Governance, Risk Management, Security Monitoring, Security Architecture Design, Data Security, Application Security, Virtual Machine Security.

Labs / Practicals:

1. Exploring Cloud Deployment and Service Models
2. Simulating Public, Private, and Hybrid Clouds
3. Virtualization of CPU and Memory
4. Network and Storage Virtualization
5. Deployment of a Layered Cloud Architecture
6. Resource Provisioning and Scaling
7. Deploying a Web Application
8. Simulating Security Attacks and Defenses in a Cloud VM
9. Implementing Access Control and Encryption on Cloud Storage

References:

[1] R. Buyya, C. Vecchiola, and T. Selvi, Mastering Cloud Computing: Foundations and Applications Programming. New Delhi, India: McGraw-Hill Education, 2013.
[2] A. T. Velte, T. J. Velte, and R. Elsenpeter, Cloud Computing: A Practical Approach. New Delhi, India: McGraw-Hill Education, 2010.
[3] K. Hwang, G. C. Fox, and J. J. Dongarra, Distributed and Cloud Computing: From Parallel Processing to the Internet of Things. Burlington, MA, USA: Morgan Kaufmann, 2012.
[4] T. Erl, R. Puttini, and Z. Mahmood, Cloud Computing: Concepts, Technology & Architecture. Boston, MA, USA: Pearson Education, 2013.
[5] G. Reese, Cloud Application Architectures: Building Applications and Infrastructure in the Cloud. Sebastopol, CA, USA: O'Reilly Media, 2009.