MK UNIVERSITY

PATAN, GUJARAT

ESTABLISHED BY THE GUJARAT GOVT. **RECOGNIZED BY UGC UNDER SECTION 2(f) OF UGC ACT,1956**

MK University, Patan
Faculty of Engineering Technology,
Department of Computer Science Engineering (CSE)

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-I										
SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	Tutorial	PRACTICAL (HRS.)/ WEEK	CREDITS	EXAMINATION		TOTAL MARKS
								INTERNAL	EXTERNAL	
1	MAJOR	BTCSE101	Engineering Mathematics-I	4	0	0	4	40	60	100
2	MAJOR	BTCSE102	Introduction to Computer Programming (Python/C)	4	0	0	4	40	60	100
3	MAJOR	BTCSE103	Engineering Physics	4	0	0	4	40	60	100
4	MINOR	BTCSE104	Data Science	4	0	0	4	40	60	100
5	MINOR	BTCSE105	Electronics	4	0	0	4	40	60	100
6	VAC	BTCSE106	Communication Skills in – I	2	0	0	2	0	50	50
TOTAL				22	0	0	22	200	350	550

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-II									
SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/W EEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERNAL	EXTERNAL	
1	MAJOR	BTCSE201	Engineering Mathematics-II	4	0	4	40	60	100
2	MAJOR	BTCSE202	Data Structures & Algorithms	4	0	4	40	60	100
3	MAJOR	BTCSE203	Digital Logic Design	4	1	5	40	60	100
4	MINOR	BTCSE204	Engineering Graphics & Design	4	1	5	40	60	100
5	MINOR	BTCSE205	Management	4	0	4	40	60	100
6	VAC	BTCSE206	Environmental Science	2	0	2	0	50	50
TOTAL				22	2	24	200	350	550

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-III									
SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/ WEEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERNAL	EXTERNAL	
1	MAJOR	BTCSE301	Discrete Mathematics	4	0	4	40	60	100
2	MAJOR	BTCSE302	Object-Oriented Programming (Java/C++)	4	1	5	40	60	100
3	MAJOR	BTCSE303	Computer Organization & Architecture	4	0	4	40	60	100
4	MAJOR	BTCSE304	Database Management Systems	4	1	5	40	60	100
5	MINOR	BTCSE305	Economics for Engineers	4	0	4	40	60	100
6	SEC	BTCSE306	Software Tool: Git	0	2	2	50	0	50
TOTAL				20	4	24	250	300	550

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-IV									
SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/ WEEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERNAL	EXTERNAL	
1	MAJOR	BTCSE401	Operating Systems	4	0	4	40	60	100
2	MAJOR	BTCSE402	Theory of Computation	4	0	4	40	60	100
3	MAJOR	BTCSE403	Microprocessors & Microcontrollers	4	1	5	40	60	100
4	MAJOR	BTCSE404	Advanced Data Structures	4	1	5	40	60	100
5	MINOR	BTCSE405	Introduction to IoT	4	0	4	40	60	100
6	I	BTCSE406	Indian Constitution	2	0	2	0	50	50
TOTAL				22	2	24	200	350	550

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-V

SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/WEEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERNAL	EXTERNAL	
1	MAJOR	BTCSE501	Computer Networks	4	1	5	40	60	100
2	MAJOR	BTCSE502	Design & Analysis of Algorithms	4	0	4	40	60	100
3	MAJOR	BTCSE503	Software Engineering	4	1	5	40	60	100
4	MAJOR	BTCSE504	Computer Graphics	4	0	4	40	60	100
5	MINOR	BTCSE505	Cloud Computing Fundamentals	4	0	4	40	60	100
6	SEC	BTCSE506	Mini-Project	0	2	2	50	0	50
TOTAL				20	4	24	250	300	350

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-VI

SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/WEEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERNAL	EXTERNAL	
1	MAJOR	BTCSE601	Compiler Design	4	0	4	40	60	100
2	MAJOR	BTCSE602	Cryptography & Network Security	4	1	5	40	60	100
3	MAJOR	BTCSE603	Advanced Computer Architecture	4	0	4	40	60	100
4	MAJOR	BTCSE604	Natural Language Processing	4	0	4	40	60	100
5	MINOR	BTCSE605	Distributed Systems	4	1	5	40	60	100
6	SEC	BTCSE606	Aptitude & Career Skills	0	2	2	50	0	50
TOTAL				20	4	24	250	300	550

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-VII									
SR NO.	COURSE TYPE	COURSE CODE	COURSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/ WEEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERNAL	EXTERNAL	
1	MAJOR	BTCSE701	Artificial Intelligence	4	0	4	40	60	100
2	MINOR	BTCSE702	Machine Learning	4	0	4	40	60	100
3	MAJOR	BTCSE703	Big Data Analytics	4	1	5	40	60	100
4	MINOR	BTCSE704	Cybersecurity	4	1	5	40	60	100
5	Multidisciplinary (MD)	BTCSE705	Entrepreneurship & Product Management	4	0	4	40	60	100
6	SEC	BTCSE706	Project Phase-I	0	2	2	50	00	50
TOTAL				20	4	24	250	300	550

B. TECH (COMPUTER SCIENCE ENGINEERING) SEM-VIII									
SR NO.	COURSE TYPE	COURSECODE	CORSE NAME	LECTURE (HRS.)/ WEEK	PRACTICAL (HRS.)/ WEEK	CREDITS	EXAMINATION		TOTAL MARKS
							INTERN AL	EXTERNAL	
1	MAJOR	BTCSE801	Advanced Algorithms	4	0	4	40	60	100
2	MAJOR	BTCSE802	Parallel Computing	4	1	5	40	60	100
4	MAJOR	BTCSE803	Robotics	4	1	5	40	60	100
5	SEC	BTCSE804	Project Phase-II	0	10	10	100	100	200
TOTAL				12	12	24	220	280	500

SUBJECT CODE: BTCSE101

SUBJECT NAME: ENGINEERING MATHEMATICS-I

Course Objective:

- The concept of rank of a matrix which is used to know the consistency of system of linear equations and also to find the eigen vectors of a given matrix.
- Finding maxima and minima of functions of several variables.
- Applications of first order ordinary differential equations. (Newton's law of cooling, Natural growth and decay)
- How to solve first order linear, nonlinear partial differential equations and also method of separation of variables technique to solve typical second order partial differential equations.
- Solving differential equations using Laplace Transforms.

Course Outcomes: At the end of the course students shall be able to

CO1	The concept of rank of a matrix which is used to know the consistency of system of linear equations and also to find the eigen vectors of a given matrix
CO2	Finding maxima and minima of functions of several variables
CO3	Applications of first order ordinary differential equations
CO4	How to solve first order linear, nonlinear partial differential equations and also method of separation of variables technique to solve typical second order partial differential equations

Unit	Content	Credit	Weightage
I	Matrices Introduction, types of matrices-symmetric, skew-symmetric, Hermitian, skew-Hermitian, orthogonal, unitary matrices. Rank of a matrix - echelon form, normal form, consistency of system of linear equations (Homogeneous and Non-Homogeneous). Eigen values and Eigen vectors and their properties (without proof), Cayley-Hamilton theorem (without proof), Diagonalization.	1	25%
II	Functions of Several Variables Limit continuity, partial derivatives and total derivative. Jacobian-Functional dependence and independence. Maxima and minima and saddle points, method of Lagrange multipliers, Taylor's theorem for two variables.	1	25%
III	Ordinary Differential Equations First order ordinary differential equations: Exact, equations reducible to exact form. Applications of first order differential equations - Newton's law of cooling, law of natural growth and decay. Linear differential equations of second and higher order with constant coefficients: Non-homogeneous term of the type f(x) = eax, sinax, cosax, xn , eax V and xn V. Method of variation of parameters.	1	25%

IV	Partial Differential Equations Introduction, formation of partial differential equation by elimination of arbitrary constants and arbitrary functions, solutions of first order Lagrange's linear equation and non-linear equations, Charpit's method, Method of separation of variables for second order equations and applications of PDE to one dimensional (Heat equation).	1	25%

TEXT BOOKS:

1. Higher Engineering Mathematics by B V Ramana ., Tata McGraw Hill.
2. Higher Engineering Mathematics by B.S. Grewal, Khanna Publishers.
3. Advanced Engineering Mathematics by Kreyszig, John Wiley & Sons.

REFERENCE BOOKS:

i)Advanced Engineering Mathematics by R.K Jain & S R K Iyenger, Narosa Publishers. ii)Advanced Engineering Mathematics by Michael Green Berg, Pearson Publishers . iii)Engineering Mathematics by N.P Bali and Manish Goyal.

SUBJECT CODE: BTCSE102

SUBJECT NAME: INTRODUCTION TO COMPUTER PROGRAMMING - C

Course Objective:

1. To express algorithms and draw flowcharts in a language independent manner.
2. To teach how to write modular, efficient and readable C programs
3. To impart knowledge in creating and using Arrays of the C data types.
4. To describe the techniques for creating program modules in C using functions and recursive functions.
5. To demonstrate creation of derived data types and perform operations on files.
6. To familiarize pointers and dynamic memory allocation functions to efficiently solve problems

Course Outcomes: At the end of the course students shall be able to

CO1	Write, compile and debug programs in C language
CO2	Use different data types in a computer program
CO3	Design programs involving decision structures, loops, arrays and functions
CO4	Identify the difference between call by value and call by reference

Unit	Content	Credit	Weightage
I	Introduction to the C Language – Algorithm, Pseudo code, Flow chart, Background, C Programs, Identifiers, Data Types, Variables, Constants, Input / Output, Operators (Arithmetic, relational, logical, bitwise etc.), Expressions, Precedence and Associatively, Expression Evaluation, Type conversions.	1	25%
II	Statements- Selection Statements(making decisions) – if and switch statements, Repetition statements (loops)-while, for, do-while statements, Loop examples, other statements related to looping – break, continue, go to, Simple C Program examples.	1	25%
III	Functions- Introduction to Structured Programming, Functions- basics, user defined functions, inter function communication(call by value, call by reference), Standard functions. Storage classes-auto, register, static, extern, scope rules, arrays to functions, recursive functions, example C programs.	1	25%
IV	Arrays– Basic concepts, one-dimensional arrays, two – dimensional arrays, multidimensional arrays, C programming examples Pointers – Introduction (Basic Concepts), pointers to pointers, compatibility, Pointer Applications, Arrays and Pointers, Pointer Arithmetic, memory allocation functions, array of pointers, pointers to void, pointers to functions, command –line arguments, Introduction to structures and unions.	1	25%

TEXT BOOKS:

1. Computer Science: A Structured Programming Approach Using C, B.A.Forouzan and R.F. Gilberg, Third Edition, Cengage Learning.

2. The C Programming Language by Brian Kernighan and Dennis Ritchie 2nd edition

REFERENCE BOOKS:

1. Let Us C Yashavant kanetkar BPB.
2. Absolute beginner's guide to C, Greg M. Perry, Edition 2, Publisher: Sams Pub., 1994.
3. Computer Programming and Data Structures by E Balagurusamy, Tata McGraw Hill.

SUBJECT CODE: BTCSE103

SUBJECT NAME: ENGINEERING PHYSICS

Course Outcomes: At the end of the course students shall be able to

CO1	Gain the knowledge on the basic concepts of oscillations exhibited by various systems in nature
CO2	Study the basic concepts of light through interference and diffraction
CO3	Explore band structure of the solids and classification of materials
CO4	Compare dielectric and magnetic properties of the materials and enable them to design and apply in different fields

Unit	Content	Credit	Weightage
I	HARMONIC OSCILLATIONS : Introduction to harmonic oscillators, simple harmonic oscillator: equation of motion and its solution (complex exponential method), damped harmonic oscillator: equation of motion and its solution, over, critical and lightly-damped oscillators; energy decay in damped harmonic oscillator, Quality factor (qualitative), forced damped harmonic oscillator: equation of motion and its solution.	1	25%
II	WAVEOPTICS: Interference- Introduction, Superposition of waves, interference of light by division of wave front-interference of reflected light in thin films, interference of light by division of amplitude Newton’s rings, Diffraction- difference between Fresnel and Fraunhofer diffraction, Fraunhofer diffraction due to single slit, Diffraction grating- Grating spectrum and resolving power.	1	25%
III	INTRODUCTION TO SOLIDS: Free electron theory (Classical & Quantum): Assumptions, Merits and drawbacks, Fermi level, Density of states, Periodic potential, Bloch’s theorem, Kronig – Penny modal, E – K diagram, Effective mass, Origin of energy bands in solids, Classification of materials: Metals, semi-conductors and insulators.	1	25%
IV	Dielectrics: Introduction, Types of polarizations (Electronic and Ionic) and calculation of their polarizabilities, internal fields in a solid, Clausius-Mossotti relation. Magnetism: Introduction, Bohr magneton, classification of dia, para and ferro magnetic materials on the basis of magnetic moment, Properties of anti-ferro and ferri magnetic materials, Hysteresis curve based on Domain theory of ferro magnetism, Soft and hard magnetic materials.	1	25%

TEXT BOOKS:

1. Engineering Physics by Arumugam, Anuradha publications.

2. Engineering Physics- B.K. Pandey, S. Chaturvedi, Cengage Learning.

REFERENCES:

1. Engineering Physics – R.K. Gaur and S.L. Gupta, DhanpatRai Publishers.
2. Engineering Physics, S Mani Naidu- Pearson Publishers.
3. Engineering physics 2nd edition –H.K. Malik and A.K. Singh.
4. Engineering Physics – P.K. Palaniswamy, Scitech publications.
5. Physics by Resnick and Haliday.

SUBJECT CODE: BTCSE104

SUBJECT NAME: DATA SCIENCE

Course Objective:

To Impart knowledge about basic computer fundamentals and programming languages for data science.
To Impart knowledge about mathematical and statistical methods for data analysis.
To Empower students with data visualization techniques and tools.
To impart knowledge about the basics of data management and Business Theory.
To impart knowledge about various machine learning techniques used for data analysis.
To enable students to develop data-based machine learning models for solving real-word applications.
To enable students to gain practical experience in programming languages and statistical and machine learning tools for data sciences.

Course Outcomes: At the end of the course students shall be able to

CO1	Apply basic data cleaning techniques to prepare data for analysis
CO2	Demonstrate proficiency in using appropriate tools and technology to collect, process, transform, summarize, and visualize data
CO3	Apply various machine learning algorithms in data-based decision-making applications, and draw accurate and useful conclusions through data analysis
CO4	Demonstrate some skills in data retrieval using Structured Query language (SQL)

Unit	Content	Credit	Weightage
I	History of computer, Basic Computer hardware, input and output devices, Basic computer architecture, input output devices, memory and CPU, networking of machines (overview of LAN, MAN, WAN, Internet, Wifi etc), types of computer (workstation, desktop, Smartphone, embedded system, etc.), Overview of Software (system software and application software with examples (mention names only)), Definition of Operating System and functions (mention names of some popular operating systems like Windows, Linux, Android, etc).	1	25%
II	Bit, Byte and Word, Number System (Base, Binary, Decimal, Octal, Hexadecimal), Conversion of number systems, Boolean logic (Boolean Gates), Boolean operators (OR, AND and NOT), ASCII code, Concept of Algorithm and Flowchart.	1	25%
III	Basics of Python programming (with a simple 'hello world' program, process of writing a program, running it, and print statement), Concept of class and object, Data-types (integer, float, string), notion of a variable, Operators (assignment, logical, arithmetic etc.),	1	25%

	accepting input from console, conditional statements (If else and Nested If else), Collections (List, Tuple, Sets and Dictionary), Loops (For Loop, While Loop & Nested Loops), iterator, string and fundamental string operations (compare, concatenation, sub-string etc.), Function, recursion.		
IV	Overview of linear and nonlinear data structure (definition, schematic view and difference), array (1D, 2D and its relation with matrix, basic operations: access elements using index, insert, delete, search), stack (concept of LIFO, basic operations: Push, Pop, peek, size), queue (concept of FIFO, basic operations: Enqueue, Dequeue, peek, size), use of List methods in Python for basic operations on array, stack and queue, overview of NumPy library and basic array operations (arrange, shape(), ndim(), dtype() etc.), binary tree (definition and schematic view only) .	1	25%

SUBJECT CODE: BTCSE105

SUBJECT NAME: ELECTRONICS

Course Outcomes: At the end of the course students shall be able to

CO1	The role of key electronic components and subsystems in the data acquisition pipeline.
CO2	Analyse simple analog and digital electronic circuits relevant to sensor interfacing and signal conditioning.
CO3	Design and implement basic data acquisition systems using microcontroller boards and common sensors
CO4	Evaluate the trade-offs in resolution, sampling rate, and power consumption in electronic data systems.

Unit	Content	Credit	Weightage
I	**Introduction to Electronics in the Data Pipeline** • The role of electronics in the data science workflow: **Sensing → Acquisition → Processing → Analysis**. • Overview of system components: Sensors, signal conditioners, Analog-to-Digital Converters (ADCs), microcontrollers/embedded processors. • Core concepts: Voltage, current, power, energy (crucial for battery-powered IoT/edge devices).	1	25%
II	**Semiconductor Devices for Computing & Sensing** • Diodes: Basic operation, LEDs (output indicators). • **Transistors (BJT & MOSFET):** As a switch (fundamental building block of digital logic and memory). Introduction to amplification. • Operational Amplifiers (Op-amps): Basic configurations (inverting, non-inverting) for signal conditioning (amplification, filtering).	1	25%

III	**Digital Electronics Foundations** • Logic Gates (AND, OR, NOT, NAND, NOR) and Boolean algebra. • Combinational circuits: Multiplexers, Decoders, Adders. • Sequential circuits: Flip-flops (SR, D), Registers, and basic counter concepts. • **Microcontroller/System-on-Chip (SoC) Introduction:** Block diagram (CPU, Memory, I/O ports, ADC, DAC). Contrast with general-purpose CPUs.	1	25%
IV	**Data Acquisition and Interface Electronics** • **Sensors & Transducers:** Types (Temperature, Light, Motion, Image), analog output. • **Signal Conditioning:** Noise, filtering (passive RC filters), amplification for ADC compatibility. • **Analog-to-Digital Conversion (ADC):** Sampling, quantization, resolution, sampling rate (Nyquist Theorem). • **Interfacing Protocols:** Serial Communication (UART, I2C, SPI) - the "language" for sensor data transfer.	1	25%

Suggested Textbooks & Resources

1. *The Art of Electronics* by Horowitz & Hill (Reference).
2. *Arduino Project Handbook* by Mark Geddes (Practical Guide).
3. *Data Acquisition Systems* by M. G. K. Rao.
4. Online Resources: Arduino Documentation, Analog Devices Tutorials, Spoken Tutorial on Circuit Simulation.

Practical / Lab Component (Aligns with NEP's Learning-by-Doing)

Students will use prototyping boards (like Arduino or Raspberry Pi Pico) and simulation software (like Tinkercad/Proteus).

L1: Basic measurements and use of Digital Multimeter (DMM).
L2: Interfacing LEDs and switches with a microcontroller.
L3: Reading analog sensor data (e.g., potentiometer, temperature sensor LM35) and displaying it.
L4: Programming a simple data logger with time-stamping.
L5: Serial communication between microcontroller and PC (Python script for data visualization).
L6: Mini-Project: Build a simple sensing station (e.g., for light/temperature) that logs data to a CSV file for later analysis.

SEMESTER-II

COURSE CODE: BTCSE201

COURSE NAME: ENGINEERING MATHEMATICS-II

Course Objective

To equip computer science students with advanced mathematical tools in linear algebra, probability, and discrete mathematics that form the theoretical foundation for algorithms, machine learning, cryptography, and computational complexity. The course emphasizes computational thinking and algorithmic applications of mathematical concepts.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	**Apply** linear algebra concepts (eigenvalues, SVD) to problems in computer graphics, data compression, and machine learning.
CO2	**Formulate** and **solve** probability models for analyzing randomized algorithms, network reliability, and performance evaluation.
CO3	**Model** computational problems using graph theory and solve basic graph algorithms relevant to networks and data structures.
CO4	**Implement** numerical methods for solving mathematical problems computationally with error analysis awareness.

Unit	Content	Credit	Weightage
I	**Advanced Linear Algebra** • **Vector Spaces:** Definition, subspaces, linear independence, basis, dimension • **Linear Transformations:** Matrix representation, kernel, image, rank-nullity theorem • **Eigenvalues and Eigenvectors:** Computation, properties, diagonalization • **Singular Value Decomposition (SVD):** Geometric interpretation, applications in data science • **Matrix Factorization:** LU, QR decompositions (algorithmic perspective)	1	25%

II	**Probability Theory for Computer Science**
	• **Probability Basics:** Axioms, conditional probability, Bayes' theorem
	• **Random Variables:** Discrete and continuous, PMF/PDF, CDF
	• **Important Distributions:**
	  ○ Discrete: Bernoulli, Binomial, Poisson, Geometric
	  ○ Continuous: Uniform, Normal, Exponential
	• **Expectation and Variance:** Properties, moments
	• **Joint Distributions:** Covariance, correlation, independence
	• **Law of Large Numbers & Central Limit Theorem:** Conceptual understanding
		1	25%

III	**Discrete Mathematics** • **Graph Theory:** Basic terminology, types of graphs, connectivity • **Special Graphs:** Trees, bipartite graphs, planar graphs • **Graph Algorithms:** Shortest path (conceptual), graph coloring • **Combinatorics:** Counting principles, permutations, combinations • **Recurrence Relations:** Formulation, solving using characteristic equations	1	25%
IV	**Numerical Methods for CS Applications** • **Numerical Linear Algebra:** Solving linear systems (Gaussian elimination, iterative methods) • **Numerical Integration:** Trapezoidal rule, Simpson's rule • **Root Finding:** Bisection method, Newton-Raphson method • **Error Analysis:** Round-off error, truncation error, stability	1	25%

Textbooks:

1. Linear Algebra and Its Applications by Gilbert Strang (for Module 1)
2. Introduction to Probability by Dimitri P. Bertsekas and John N. Tsitsiklis (for Module 2)
3. Discrete Mathematics and Its Applications by Kenneth H. Rosen (for Module 3)

Reference books:

1. Numerical Recipes by Press et al. (for Module 4)
2. Convex Optimization by Boyd and Vandenberghe (for Module 5)
3. Mathematics for Computer Science (MIT OpenCourseWare)

Digital Resources:

- 3Blue1Brown YouTube series (linear algebra, calculus)
- Khan Academy probability and linear algebra modules
- Jupyter notebooks for computational examples

COURSE CODE: BTCSE202
COURSE NAME: DATA STRUCTURES & ALGORITHMS (DSA)

Course Objective

To develop a deep understanding of fundamental and advanced data structures, their implementations, and associated algorithms. The course emphasizes analyzing time/space complexity, selecting appropriate data structures for problem-solving, and applying algorithmic design paradigms to real-world computing challenges.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	**Analyze** the time and space complexity of algorithms using asymptotic notations and recurrence relations.
CO2	**Implement** fundamental and advanced data structures using appropriate programming languages.
CO3	**Select** and justify appropriate data structures for solving specific computational problems.
CO4	**Design** and implement algorithms using standard paradigms (divide & conquer, greedy, dynamic programming).

Unit	Content	Credit	Weightage
I	**Foundations and Analysis** • **Algorithm Analysis:** Asymptotic notations (Big-O, Big-Ω, Big-θ), best/worst/average cases • **Complexity Analysis:** Time-space tradeoffs, amortized analysis, recurrence relations • **Abstract Data Types (ADTs):** Specification vs. implementation • **Review of C/C++/Python essentials:** Pointers, recursion, memory management	1	25%
II	**Fundamental Data Structures** • **Arrays:** Operations, dynamic arrays, applications • **Linked Lists:** o Singly, doubly, circular linked lists o Operations, memory representation o Applications: polynomial representation, sparse matrices • **Stacks:** o Array and linked list implementations o Applications: Expression evaluation, recursion, backtracking	1	25%
III	**Trees and Advanced Structures** • **Trees:** Terminology, binary trees, properties • **Binary Search Trees (BST):** Operations (insert, delete, search), traversal (in/pre/post order, level order) • **Balanced Trees:** o AVL Trees: Rotations, insertion, deletion o Red-Black Trees (concepts) • **Heaps:** Min-heap, max-heap, heap operations, heap sort • **Tries:** Structure, applications in autocomplete • **B-Trees:** Structure, applications in databases	1	25%

IV	**Hashing and Dictionaries** • **Hash Tables:** Direct addressing, hash functions • **Collision Resolution:** Chaining, open addressing (linear/quadratic probing, double hashing) • **Performance Analysis:** Load factor, rehashing • **Applications:** Dictionaries, symbol tables, caching	1	25%

Textbooks:

1. Introduction to Algorithms by Cormen, Leiserson, Rivest, Stein (CLRS)
2. Data Structures and Algorithms in Python/C++ by Goodrich, Tamassia, Goldwasser
3. Algorithms by Sedgewick and Wayne

Reference Books:

1. The Algorithm Design Manual by Steven Skiena
2. Programming Pearls by Jon Bentley
3. Cracking the Coding Interview by Gayle Laakmann McDowell

Online Platforms:

- Visualization: VisuAlgo, Algorithm Visualizer
- Practice: LeetCode, HackerRank, Codeforces
- MOOCs: MIT OpenCourseWare (6.006), Coursera (Stanford Algorithms)

Tools & Software:

- Python with libraries (timeit, memory_profiler, matplotlib for plots)
- C++ STL/Java Collections Framework
- Jupyter Notebooks for interactive learning

COURSE CODE: BTCSE203

COURSE NAME: DIGITAL LOGIC DESIGN (DLD)

Course Objective

To provide a comprehensive understanding of digital systems design fundamentals, from basic logic gates to complex sequential circuits and memory elements. The course aims to develop the ability to analyze, design, and implement digital circuits that form the foundation of computer architecture, embedded systems, and digital hardware.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	**Apply** number systems, codes, and binary arithmetic to solve digital computation problems.
CO2	**Analyze** and **simplify** Boolean functions using algebraic methods and K-maps.
CO3	**Design** and **implement** combinational circuits for arithmetic operations, data routing, and code conversion.
CO4	**Design** sequential circuits including counters, registers, and finite state machines.

Unit	Content	Credit	Weightage
I	**Number Systems and Codes** • **Number Systems:** Binary, Octal, Hexadecimal, conversions • **Complements:** 1's, 2's, 9's, 10's complements • **Binary Arithmetic:** Addition, subtraction, multiplication, division • **Binary Codes:** BCD, Gray code, Excess-3, ASCII, Unicode • **Error Detection & Correction:** Parity bits, Hamming codes	1	25%
II	**Boolean Algebra and Logic Gates** • **Boolean Algebra:** Postulates, theorems, De Morgan's laws • **Logic Gates:** AND, OR, NOT, NAND, NOR, XOR, XNOR - symbols, truth tables • **Gate-level Implementation:** SOP, POS forms • **Universal Gates:** Implementation using NAND/NOR only • **Integrated Circuits:** SSI, MSI, LSI, VLSI concepts, TTL/CMOS families	1	25%
III	**Combinational Logic Design (12 Hours)** • **Minimization Techniques:** o Karnaugh Maps (2-5 variables) o Quine-McCluskey method o Don't care conditions • **Combinational Circuits:** o Adders (Half, Full, Ripple carry, Carry look-ahead) o Subtractors o Comparators o Multiplexers (MUX) and Demultiplexers (DEMUX) o Encoders and Decoders (Binary, BCD to 7-segment) • **Code Converters:** Binary to Gray, BCD to Excess-3, etc. • **Hazards:** Static, dynamic hazards and elimination	1	25%

IV	**Sequential Logic Design** • **Latches:** SR latch, D latch, gated latches • **Flip-flops:** SR, JK, D, T, Master-slave, edge-triggered • **Flip-flop Conversions:** One type to another • **Registers:** Shift registers (SISO, SIPO, PISO, PIPO), Universal shift register • **Counters:** ○ Asynchronous (Ripple) counters ○ Synchronous counters ○ Mod-N counters, up/down counters ○ Ring counter, Johnson counter • **Finite State Machines (FSM):** Mealy and Moore models	1	25%

Textbooks:

1. Digital Design by M. Morris Mano and Michael D. Ciletti
2. Fundamentals of Digital Logic with Verilog Design by Stephen Brown and Zvonko Vranesic
3. Digital Logic and Computer Design by M. Morris Mano

Reference Books:

1. Digital Electronics by R.P. Jain
2. Verilog HDL: A Guide to Digital Design and Synthesis by Samir Palnitkar
3. Introduction to Logic Design by Alan B. Marcovitz

Online Platforms:

- Simulation Tools: Logisim, Digital Works, CircuitVerse
- HDL Simulators: EDA Playground, ModelSim (student edition)
- MOOCs: NPTEL (Digital Circuits), Coursera (Digital Systems)

Laboratory Component

Tools: Digital Trainer Kits, ICs, Breadboards, Multimeters, Simulation Software (Logisim/Proteus/Quartus), Verilog/VHDL compiler

List of Experiments:

1. **Lab 1:** Verification of truth tables for logic gates
2. **Lab 2:** Implementation of Boolean functions using logic gates
3. **Lab 3:** Design and implementation of adders/subtractors
4. **Lab 4:** Implementation of code converters
5. **Lab 5:** Design using multiplexers and demultiplexers
6. **Lab 6:** Flip-flop characteristics and conversions
7. **Lab 7:** Design of synchronous/asynchronous counters
8. **Lab 8:** Shift register applications
9. **Lab 9:** Finite State Machine implementation
10. **Lab 10:** HDL programming: Basic gate implementation
11. **Lab 11:** HDL programming: Combinational circuit (adder/MUX)
12. **Lab 12:** Mini-project: Digital clock/calculator/traffic light controller

COURSE CODE: BTCSE204
COURSE NAME: ENGINEERING GRAPHICS & DESIGN

Course Objective

1. To develop the ability to communicate design ideas using standard engineering drawing practices.
2. To introduce computer-aided design (CAD) tools for creating and modifying technical drawings.
3. To foster spatial visualization and reasoning skills.
4. To relate engineering graphics principles to computer science applications such as GUI design, game environments, 3D modeling, and hardware schematics.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Apply standard engineering drawing conventions to create and interpret technical drawings.
CO2	Generate orthographic, isometric, and sectional views of objects using manual and CAD tools.
CO3	Use CAD software to produce 2D drafts and basic 3D models relevant to CS applications.
CO4	Visualize and represent objects in 3D space, improving spatial reasoning skills.

Unit	Content	Credit	Weightage
I	**Fundamentals of Engineering Drawing** • Introduction to drawing instruments, standards (BIS/ISO) • Lettering, line types, dimensioning • Geometrical constructions	1	25%
II	**Orthographic Projections** • Principles of first-angle and third-angle projection • Projection of points, lines, planes, and solids • Sectional views	1	25%
III	**Isometric and Pictorial Views** • Isometric projection of simple solids • Perspective basics	1	25%
IV	**Computer-Aided Design (CAD)** • Introduction to CAD software (e.g., AutoCAD, Fusion 360, SketchUp) • Basic 2D drafting and 3D modeling • Editing and dimensioning in CAD	1	25%

Textbooks:

- "Engineering Drawing" by N.D. Bhatt (Charotar Publishing)
- "Engineering Drawing" by P.S. Gill (SK Kataria & Sons)
- "Engineering Drawing with CAD" by M.B. Shah & B.C. Rana (Pearson)

Reference Books:

- "Technical Drawing" by Giesecke, Mitchell, Spencer, Hill, Dygdon, Novak, Lockhart (Prentice Hall)
- "Engineering Drawing and Design" by Cecil Jensen, Jay Helsel, Dennis Short (McGraw-Hill)
- "Fundamentals of Engineering Drawing" by Warren J. Luzadder & Jon M. Duff (Prentice Hall)

Digital Libraries

- Engineering Design Graphics Journal (EDGJ)
- CAD Society resources
- IEEE Computer Graphics and Applications

COURSE CODE: BTCSE205
COURSE NAME: MANAGEMENT
Course Objective

1. To understand basic management principles in technology organizations
2. To develop skills in software project management and team leadership
3. To analyze business strategies for technology products/services
4. To comprehend financial, legal, and ethical aspects of tech businesses

Course Outcomes (COs)
Upon successful completion, students will be able to:

CO1	Apply management principles to software project planning and execution
CO2	Lead and motivate technical teams effectively using appropriate leadership
CO3	Analyze technology business strategies and make informed decisions
CO4	Develop basic financial plans and budgets for IT projects

Unit	Content	Credit	Weightage
I	**Introduction to Management in Technology** • **1.1** Evolution of management thought • **1.2** Functions of management (Planning, Organizing, Leading, Controlling) • **1.3** Types of technology organizations: product vs service, startup vs enterprise • **1.4** Role of engineers in management • **1.5** Technical vs managerial career paths	1	25%
II	**Project Management for Software** • **2.1** Software Development Life Cycle (SDLC) models • **2.2** Agile methodologies (Scrum, Kanban, XP) • **2.3** Project planning, scheduling, and estimation techniques • **2.4** Risk management in software projects • **2.5** Tools: Jira, Trello, Asana, Microsoft Project	1	25%
III	**Team Management and Leadership** • **3.1** Team formation and dynamics (Tuckman's model) • **3.2** Motivation theories (Maslow, Herzberg, McClelland) • **3.3** Leadership styles for tech teams • **3.4** Conflict resolution and negotiation • **3.5** Diversity and inclusion in tech workplaces	1	25%
IV	**Technology Strategy and Innovation** • **4.1** Technology adoption lifecycle (Crossing the Chasm) • **4.2** Competitive analysis and SWOT in tech • **4.3** Innovation management and R&D • **4.4** Intellectual property in software (patents, copyrights, open source) • **4.5** Digital transformation strategies	1	25%

Textbooks:

1. "The Mythical Man-Month" by Frederick P. Brooks Jr.

2. "Peopleware: Productive Projects and Teams" by Tom DeMarco & Timothy Lister
3. "Software Project Management" by Bob Hughes & Mike Cotterell
4. "The Lean Startup" by Eric Ries
5. "Crossing the Chasm" by Geoffrey A. Moore

Reference Books:

1. "The Phoenix Project" by Gene Kim, Kevin Behr, George Spafford
2. "Project Management for Engineering, Business and Technology" by John M. Nicholas & Herman Steyn
3. "Engineering Management: Challenges in the New Millennium" by William B. Rouse
4. "Managing the Unmanageable: Rules, Tools, and Insights for Managing Software People and Teams" by Mickey W. Mantle & Ron Lichty

Online Resources:

1. Harvard Business Review (Tech Management section)
2. MIT Sloan Management Review
3. Project Management Institute (PMI) resources
4. Agile Alliance materials
5. Case studies from: Stanford, IIMs, ISB

SEMESTER-III

COURSE CODE: BTCSE301
COURSE NAME: DISCRETE MATHAMATICS
Course Objective

1. To develop mathematical reasoning and proof-writing skills
2. To understand and apply discrete structures in CS contexts
3. To master formal logic and its applications in computing
4. To analyze counting problems and probability in algorithms

Course Outcomes (COs)
Upon successful completion, students will be able to:

CO1	Construct formal proofs using various proof techniques
CO2	Apply logic to formulate and solve computational problems
CO3	Solve counting problems using combinatorial methods
CO4	Model relationships and networks using graph theory

Unit	Content	Credit	Weightage
I	**Foundations - Logic and Proofs** • **1.1 Propositional Logic** ○ Propositions, connectives, truth tables ○ Logical equivalences, tautologies, contradictions ○ Normal forms (DNF, CNF) • **1.2 Predicate Logic** ○ Quantifiers (∀, ∃), nested quantifiers ○ Rules of inference **1.3 Proof Techniques** ○ Direct, contrapositive, contradiction ○ Mathematical induction, strong induction ○ Structural induction (on trees, expressions) ○ Proof by cases, counterexamples	1	25%
II	**Sets, Relations, and Functions** • **2.1 Set Theory** ○ Operations, laws, cardinality ○ Power sets, Cartesian products ○ Infinite sets, countable/uncountable sets (Cantor's diagonalization) • **2.2 Relations** ○ Properties (reflexive, symmetric, transitive, equivalence) ○ Equivalence relations and partitions ○ Partial orders, Hasse diagrams, lattices • **2.3 Functions** ○ Injections, surjections, bijections ○ Inverse functions, composition ○ Floor, ceiling, modulo functions ○ Growth of functions (Big-O, Ω, Θ notation)	1	25%

III	**Combinatorics and Discrete Probability** • **3.1 Counting Principles** o Sum and product rules o Inclusion-exclusion principle o Pigeonhole principle and applications • **3.2 Permutations and Combinations** o With/without repetition o Binomial coefficients, Pascal's triangle o Combinatorial identities **3.3 Advanced Counting** o Recurrence relations o Generating functions o Solving recurrences (characteristic equations) o **3.4 Discrete Probability** o Finite probability spaces o Conditional probability, Bayes' theorem o Random variables, expectation, variance o Probabilistic method applications	1	25%
IV	**Graph Theory** • **4.1 Basic Concepts** o Graphs, multigraphs, pseudographs o Directed graphs, weighted graphs o Paths, cycles, connectivity • **4.2 Special Graphs** o Complete, bipartite, regular graphs o Trees and their properties o Planar graphs, Euler's formula o Graph coloring, chromatic number o **4.3 Graph Algorithms** o Graph representation (adjacency matrix/list) o Traversal (DFS, BFS) o Shortest path (Dijkstra's algorithm) o Minimum spanning trees (Prim's, Kruskal's) o Matching problems (maximum bipartite matching)	1	25%

Textbooks:

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen (7th/8th edition)
- "Concrete Mathematics" by Graham, Knuth, Patashnik
- "Discrete Mathematics for Computer Scientists" by Stein, Drysdale, Bogart
- "Mathematics for Computer Science" by Eric Lehman, F.T. Leighton, A.R. Meyer
- "Discrete Mathematics with Graph Theory" by Goodaire & Parmenter

Reference Books:

- "Introduction to Algorithms" by Cormen et al. (Appendix on mathematics)
- "The Art of Computer Programming, Vol. 1" by Donald Knuth
- "A Textbook of Discrete Mathematics" by Swapan Kumar Sarkar

Online Resources:

- MIT OpenCourseWare: 6.042 Mathematics for Computer Science
- Coursera: Discrete Mathematics Specialization (UC San Diego)
- NPTEL: Discrete Mathematics courses
- ProofWiki: Online compendium of mathematical proofs

COURSE CODE: BTCSE302
COURSE NAME: OBJECT ORIENTED PROGRAMMING (OOP)
Course Objective

1. To understand object-oriented programming principles and their advantages
2. To master C++ syntax, semantics, and standard library
3. To develop skills in designing and implementing OOP solutions
4. To understand memory management and performance considerations in C++

Course Outcomes (COs)
Upon successful completion, students will be able to:

CO1	Design and implement classes using OOP principles
CO2	Apply inheritance and polymorphism to create extensible software
CO3	Utilize templates and STL for generic programming
CO4	Implement proper memory management and exception handling

Unit	Content	Credit	Weightage

I	**C++ Fundamentals and Review of C Concepts**
	• **1.1 Introduction to C++**
	  ○ History, standards (C++98, C++11, C++14, C++17, C++20)
	  ○ Compilation process (g++, clang++)
	  ○ Basic program structure
	• **1.2 C Review in C++ Context**
	  ○ Data types, operators, control structures
	  ○ Functions, pointers, arrays
	  ○ Structures and unions
	• **1.3 C++ Enhancements over C**
	  ○ Input/output streams (cin, cout)
	  ○ References vs pointers
	  ○ Function overloading
	  ○ Default arguments, inline functions
	  ○ Namespaces
		1	25%

II	**Introduction to Object-Oriented Programming**
	• **2.1 OOP Concepts**
	  ○ Abstraction, encapsulation, inheritance, polymorphism
	  ○ Classes vs objects
	  ○ Benefits of OOP
	• **2.2 Classes and Objects**
	  ○ Class definition, access specifiers (public, private, protected)
	  ○ Member functions, constructors, destructors
	  ○ Static members, friend functions/classes
	  ○ this pointer
	• **2.3 Object Lifecycle**
		1	25%

	○ Constructors (default, parameterized, copy) ○ Destructors ○ Dynamic object creation (new, delete) ○ Object arrays		
III	**Advanced Class Features and Memory Management** • **3.1 Operator Overloading** ○ Unary and binary operators ○ Friend operators ○ Overloading I/O operators (<<, >>) ○ Type conversion operators • **3.2 Memory Management** ○ Stack vs heap allocation ○ Dynamic arrays, 2D arrays ○ Memory leaks and debugging tools (Valgrind) ○ RAII (Resource Acquisition Is Initialization) • **3.3 Advanced Class Design** ○ Constant objects and member functions ○ Mutable members ○ Nested classes, local classes	1	25%
IV	**Inheritance and Polymorphism** • **4.1 Inheritance Basics** ○ Base and derived classes ○ Access control (public, private, protected inheritance) ○ Constructors and destructors in inheritance ○ Function overriding • **4.2 Types of Inheritance** ○ Single, multiple, multilevel, hierarchical, hybrid ○ Virtual base classes ○ Diamond problem and solution • **4.3 Polymorphism** ○ Virtual functions, pure virtual functions ○ Abstract classes and interfaces ○ Virtual destructors ○ Late binding vs early binding ○ vtable and vpointer concepts	1	25%

Textbooks:

- "The C++ Programming Language" by Bjarne Stroustrup (4th edition)
- "Object-Oriented Programming in C++" by Robert Lafore
- "C++ Primer" by Stanley B. Lippman, Josée Lajoie, Barbara E. Moo

Reference Books:

- "Effective C++" and "Effective Modern C++" by Scott Meyers
- "Design Patterns: Elements of Reusable Object-Oriented Software" by GoF
- "C++ Concurrency in Action" by Anthony Williams
- "Programming: Principles and Practice Using C++" by Bjarne Stroustrup
- Data Structures and Algorithms in C++" by Michael T. Goodrich et al.

Online Resources:

- cppreference.com (Official C++ reference)
- learncpp.com (Free tutorials)
- **C++ Core Guidelines**
- **ISO C++ Standard documents**
- **Stack Overflow C++ tag**
- **YouTube:** CppCon, Meeting C++ conferences

Laboratory Experiments

Lab 1: Basic C++ programs with classes and objects
Lab 2: Operator overloading and friend functions
Lab 3: Inheritance and polymorphism implementations
Lab 4: File handling and exception handling
Lab 5: Template programming
Lab 6: STL containers and algorithms
Lab 7: Smart pointers and modern C++ features
Lab 8: Multithreading basics
Lab 9-10: Mini-project implementing design patterns
Lab 11-12: Comprehensive project (Library Management, Banking System, etc.)

COURSE CODE: BTCSE303
COURSE NAME: COMPUTER ORGANIZATION & ARCHITECTURE (COA)
Course Objective

This course introduces the fundamental concepts of computer organization and architecture, covering the design and operation of digital computers from gate-level to system-level. It explores processor design, memory hierarchy, I/O systems, and parallel processing, providing the foundation for understanding modern computer systems and their performance characteristics.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Analyze computer performance using various metrics
CO2	Design basic arithmetic and logic units
CO3	Explain processor data path and control unit operations
CO4	Analyze memory hierarchy and its impact on performance

Unit	Content	Credit	Weightage
I	**Introduction and Basic Computer Structure** • **1.1 Evolution of Computers** ○ Generations of computers (Vacuum tubes to VLSI) ○ Moore's Law and its implications ○ Types of computers: embedded, desktop, servers, supercomputers • **1.2 Computer Components and Functions** ○ Von Neumann architecture ○ Harvard architecture ○ CPU, memory, I/O subsystems	1	25%

	• **1.3 Performance Metrics** ○ Clock rate, CPI, MIPS, MFLOPS ○ Amdahl's Law ○ Benchmarking and performance evaluation		
II	**Data Representation and Computer Arithmetic** • **2.1 Number Systems** ○ Binary, octal, hexadecimal ○ Integer representation: signed magnitude, 1's complement, 2's complement ○ Floating-point representation: IEEE 754 standard • **2.2 Computer Arithmetic** ○ Addition/subtraction of signed numbers ○ Multiplication algorithms: Booth's algorithm ○ Division algorithms: Restoring and non-restoring ○ Floating-point arithmetic operations	1	25%
III	**Processor Design and Instruction Set Architecture** • **3.1 Instruction Set Architecture (ISA)** ○ CISC vs RISC philosophies ○ Addressing modes (immediate, direct, indirect, indexed, relative) ○ Instruction formats (zero, one, two, three address) • **3.2 MIPS Architecture (Case Study)** ○ Register set and memory organization ○ Instruction types: R-type, I-type, J-type ○ Sample MIPS programs • **3.3 CPU Organization** ○ Register organization ○ ALU design ○ Control unit: Hardwired vs microprogrammed control	1	25%
IV	**Processor Datapath and Control** • **4.1 Single-Cycle Implementation** ○ Datapath components (PC, registers, ALU, memory) ○ Control signals generation ○ Performance limitations • **4.2 Multi-Cycle Implementation** ○ Breaking instructions into multiple cycles ○ Finite state machine control • **4.3 Pipelining** ○ Basic 5-stage pipeline (IF, ID, EX, MEM, WB) ○ Pipeline hazards: structural, data, control ○ Hazard detection and resolution ○ Forwarding (bypassing) and stalling	1	25%

Textbooks:

- "Computer Organization and Design: The Hardware/Software Interface" by David A. Patterson and John L. Hennessy (MIPS/RISC-V editions)
- "Computer Architecture: A Quantitative Approach" by John L. Hennessy and David A. Patterson
- "Computer Organization and Architecture" by William Stallings

Reference Books:

- "Structured Computer Organization" by Andrew S. Tanenbaum
- "Digital Design and Computer Architecture" by David Harris and Sarah Harris
- "Computer Systems: A Programmer's Perspective" by Bryant and O'Hallaron
- "Introduction to High Performance Computing for Scientists and Engineers" by Georg Hager and Gerhard Wellein

Online Resources:

1. NPTEL Courses: Computer Organization by Prof. S. Raman (IIT Madras)
2. MIT Open Course Ware: Computation Structures
3. UC Berkeley CS61C: Great Ideas in Computer Architecture
4. Coursera: Computer Architecture by Princeton University
5. Wikipedia: CPU design articles
6. ARM Developer Documentation
7. RISC-V Specifications

COURSE CODE: BTCSE304
COURSE NAME: DATA BASE MANAGEMENT SYSTEM (DBMS)
Course Objective

This course introduces fundamental concepts of database systems, covering data modeling, database design, SQL programming, transaction management, and system architecture. Emphasis is placed on relational database theory, normalization, and practical implementation using modern DBMS technologies. The course prepares students to design, implement, and manage database systems for real-world applications.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Design database schemas using ER modeling and normalization
CO2	Implement and query databases using SQL and PL/SQL
CO3	Apply transaction management and concurrency control
CO4	Design efficient storage structures and indexing schemes

Unit	Content	Credit	Weightage
I	**Introduction to Database Systems** • **1.1 Database System Concepts** ○ Data, information, and knowledge ○ File systems vs database systems ○ Three-schema architecture (internal, conceptual, external) ○ Data independence (logical and physical) • **1.2 Database System Architecture** ○ DBMS components: Query processor, storage manager ○ Database users and administrators	1	25%

	◦ Database languages: DDL, DML, DCL • **1.3 Database Applications** ◦ Traditional applications (banking, airlines, universities) ◦ Modern applications (e-commerce, social media, IoT) ◦ Emerging trends (big data, cloud databases)		
II	**Data Models and Database Design** • **2.1 Entity-Relationship Model** ◦ Entities, attributes, relationships ◦ ER diagram notation (Chen notation, Crow's foot) ◦ Cardinality constraints (1:1, 1:N, M:N) ◦ Weak entities, composite attributes, multivalued attributes • **2.2 Enhanced ER Modelling** ◦ Specialization and generalization ◦ Aggregation ◦ Inheritance in EER • **2.3 Relational Model** ◦ Relations, tuples, attributes, domains ◦ Keys: Super, candidate, primary, foreign ◦ Relational algebra operations ▪ Basic: select, project, union, set difference, Cartesian product ▪ Additional: rename, intersection, natural join, division ◦ Relational calculus (tuple and domain)	1	25%
III	**Structured Query Language (SQL)** • **3.1 SQL Fundamentals** ◦ Data types, schema definition ◦ Basic queries: SELECT, FROM, WHERE ◦ Aggregate functions, GROUP BY, HAVING ◦ Set operations: UNION, INTERSECT, EXCEPT • **3.2 Advanced SQL** ◦ Nested subqueries (correlated and non-correlated) ◦ JOIN operations: INNER, LEFT, RIGHT, FULL OUTER ◦ Views: creation, updating, materialized views ◦ Integrity constraints: NOT NULL, UNIQUE, CHECK, DEFAULT	1	25%

	• **3.3 SQL Programming**
	  ○ PL/SQL or T-SQL basics
	  ○ Stored procedures, functions, triggers
	  ○ Cursors, exception handling
	  ○ Embedded SQL, dynamic SQL		

IV	**Database Design Theory**
	• **4.1 Functional Dependencies**
	  ○ Definition and properties
	  ○ Armstrong's axioms
	  ○ Closure of attribute sets
	  ○ Canonical cover
	• **4.2 Normalization**
	  ○ First Normal Form (1NF)
	  ○ Second Normal Form (2NF)
	  ○ Third Normal Form (3NF)
	  ○ Boyce-Codd Normal Form (BCNF)
	  ○ Higher normal forms (4NF, 5NF) overview
	• **4.3 Decomposition Algorithms**
	  ○ Lossless join decomposition
	  ○ Dependency preserving decomposition
	  ○ Synthesis algorithm for 3NF
	  ○ Decomposition algorithm for BCNF
		1	25%

Textbooks:

- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe
- "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke

Reference Books:

- "SQL and Relational Theory: How to Write Accurate SQL Code" by C.J. Date
- "Transaction Processing: Concepts and Techniques" by Jim Gray and Andreas Reuter
- "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" by Pramod J. Sadalage and Martin Fowler
- "Data Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems" by Martin Kleppmann

Online Resources:

- Stanford Online: Databases course by Jennifer Widom
- Coursera: Database series by University of Michigan
- NPTEL: Database Management Systems courses
- PostgreSQL Documentation
- MySQL Reference Manual
- MongoDB University (free courses)
- db-fiddle.com (online SQL playground)

Laboratory Experiments

- **Lab 1:** Installation and configuration of DBMS (MySQL/PostgreSQL)
- **Lab 2:** Basic SQL – DDL, DML commands

- **Lab 3:** SQL queries with aggregate functions and grouping
- **Lab 4:** Advanced SQL – Subqueries, joins, views
- **Lab 5:** PL/SQL programming – Stored procedures and functions
- **Lab 6:** Triggers and cursors implementation
- **Lab 7:** ER modeling and conversion to relational schema
- **Lab 8:** Normalization exercises
- **Lab 9:** Index creation and performance analysis
- **Lab 10:** Transaction management and concurrency control
- **Lab 11:** NoSQL database basics (MongoDB)
- **Lab 12:** Mini-project – Complete database application

COURSE CODE: BTCSE305
COURSE NAME: ECONOMICS FOR ENGINEERS
Course Objective
This course introduces fundamental economic principles and decision-making frameworks specifically tailored for engineering students. It covers microeconomics, macroeconomics, engineering economics, and technology economics, with special emphasis on applications in the technology sector. The course equips future engineers with economic thinking skills essential for project evaluation, business decisions, and technology strategy in engineering careers.

Course Outcomes (COs)
Upon successful completion, students will be able to:

CO1	Apply microeconomic principles to analyze technology markets
CO2	Perform engineering economic analysis for project evaluation
CO3	Calculate and interpret key financial metrics for tech projects
CO4	Analyze macroeconomic factors affecting technology industry

Unit	Content	Credit	Weightage
I	**Introduction to Economics for Engineers** • **1.1 Why Economics for Engineers?** o Economic decision-making in engineering projects o Cost-benefit analysis in technology development o Role of engineers in business and economic growth • **1.2 Basic Economic Concepts** o Scarcity, choice, and opportunity cost o Production possibilities frontier o Economic systems: market, command, mixed o Circular flow of economic activity • **1.3 Engineering as Economic Activity** o Value creation through technology	1	25%

	○ Engineering efficiency vs economic efficiency ○ Time value of money in engineering projects		
II	**Microeconomics for Technology Markets** • **2.1 Demand and Supply Analysis** ○ Market forces in technology products ○ Elasticity: price, income, cross elasticity ○ Technology adoption curves • **2.2 Consumer Behaviour** ○ Utility theory and indifference curves ○ Budget constraints ○ Consumer surplus in technology markets • **2.3 Production and Costs** ○ Production functions (short run vs long run) ○ Cost concepts: fixed, variable, marginal, average ○ Economies of scale in tech industry ○ Learning curves in software development • **2.4 Market Structures** ○ Perfect competition ○ Monopoly and natural monopolies in tech ○ Oligopoly (telecom, operating systems) • Monopolistic competition (app stores, SaaS) • Network effects and winner-take-all markets	1	25%
III	**Engineering Economics** • **3.1 Time Value of Money** ○ Present value, future value calculations ○ Discounting and compounding ○ Net Present Value (NPV) method ○ Internal Rate of Return (IRR) • **3.2 Investment Appraisal Techniques** ○ Payback period ○ Accounting Rate of Return (ARR) ○ Benefit-Cost Ratio (BCR) ○ Break-even analysis for tech products • **3.3 Cost Analysis for Engineering Projects** ○ Life Cycle Costing (LCC) ○ Capital vs operational expenditures (CapEx vs OpEx) ○ Depreciation methods (straight-line, declining balance) ○ Total Cost of Ownership (TCO) for IT systems	1	25%
IV	**Macroeconomics for Engineers** • **4.1 National Income Accounting**	1	25%

	◦ GDP, GNP, NNP concepts ◦ Economic growth and development ◦ Technology's contribution to GDP • **4.2 Money, Banking, and Inflation** ◦ Monetary policy and interest rates ◦ Inflation's impact on engineering projects ◦ Cryptocurrencies and digital payments • **4.3 Business Cycles** ◦ Economic fluctuations and technology investment ◦ Recessions and tech industry resilience ◦ Leading indicators for tech sector		

Textbooks:

- "Engineering Economy" by William G. Sullivan, Elin M. Wicks, and C. Patrick Koelling
- "Principles of Economics" by N. Gregory Mankiw
- "Managerial Economics and Business Strategy" by Michael Baye and Jeff Prince

Reference Books:

- "Economics for Engineers" by T. T. Sethi and A. K. Nath
- "Financial Management for Engineers" by C. M. Chang
- "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation" by Eric Ries

Online Resources:

- Khan Academy: Microeconomics and Macroeconomics courses
- Coursera: Economics for Engineers courses
- Harvard Business Review: Case studies on technology economics
- St. Louis Fed FRED: Economic data for analysis
- TechCrunch, Wired: Current technology economics articles
- Bureau of Economic Analysis (BEA) data
- World Bank Open Data

COURSE CODE: BTCSE306
COURSE NAME: SOFTWARE GIT
Course Objective

This practical course introduces version control systems with a focus on Git, covering fundamental concepts, workflows, and best practices for collaborative software development. Students will learn to manage code repositories, collaborate effectively, integrate with modern development tools, and understand industry-standard Git workflows used in professional software engineering.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Set up and configure Git for individual and team development
CO2	Implement branching strategies and resolve merge conflicts

CO3	Collaborate effectively using remote repositories and pull requests		
Unit	Content	Credit	Weightage

I	**Introduction to Version Control**
	• **1.1 Why Version Control?**
	  ○ The problem of code collaboration
	  ○ History of version control: Local → Centralized → Distributed
	  ○ Benefits: Collaboration, backup, experimentation, accountability
	• **1.2 Version Control Systems Landscape**
	  ○ CVS, SVN (Centralized)
	  ○ Git, Mercurial (Distributed)
	  ○ Comparison of different VCS
	• **1.3 Real-world Applications**
	  ○ Open-source projects (Linux kernel, Android)
	  ○ Enterprise development workflows
	  ○ Academic and research code management		

II	**Git Fundamentals**
	• **2.1 Git Architecture**
	  ○ Three states: Working directory, staging area, repository
	  ○ Three main sections: .git directory, staging area, working tree
	  ○ Snapshots, not differences
	• **2.2 Getting Started with Git**
	  ○ Installation and configuration (git config)
	  ○ Setting up identity (name, email)
	  ○ Global vs local configuration
	  ○ Initializing repositories (git init, git clone)
	• **2.3 Basic Git Commands**
	  ○ git status - Checking repository state
	  ○ git add - Staging changes
	  ○ git commit - Committing changes
	  ○ git log - Viewing history
	  ○ git diff - Comparing changes		

III	**Branching and Merging**
	• **3.1 Understanding Branches**
	  ○ What are branches? (Lightweight movable pointers)
	  ○ HEAD pointer concept
	  ○ Creating and switching branches (git branch, git checkout, git switch)		

	• **3.2 Branch Operations** ○ Listing branches ○ Creating feature branches ○ Deleting branches ○ Remote branch tracking • **3.3 Merging Strategies** ○ Fast-forward merges ○ Three-way merges ○ Merge conflicts: identification and resolution ○ git merge vs git rebase • **3.4 Advanced Branching** ○ Rebasing: cleaning up commit history ○ Interactive rebase (git rebase -i) ○ Cherry-picking commits ○ Stashing changes (git stash)		

Textbooks:

- "Pro Git" by Scott Chacon and Ben Straub (Free online at git-scm.com/book)
- "Version Control with Git" by Jon Loeliger and Matthew McCullough

Reference Books:

- "Git Pocket Guide" by Richard E. Silverman
- "Learning Git" by Anna Skoulikari
- "Git for Teams" by Emma Jane Hogbin Westby

Online Resources:

- Official Git Documentation: git-scm.com/doc
- GitHub Guides: guides.github.com
- GitLab Documentation: docs.gitlab.com
- Atlassian Git Tutorials: atlassian.com/git
- Git Immersion Tutorial: gitimmersion.com
- Learn Git Branching: learngitbranching.js.org (Interactive)
- Oh My Git!: ohmygit.org (Game for learning Git)

COURSE CODE: BTCSE304
COURSE NAME: DATA BASE MANAGEMENT SYSTEM (DBMS)
Course Objective

This course introduces fundamental concepts of database systems, covering data modeling, database design, SQL programming, transaction management, and system architecture. Emphasis is placed on relational database theory, normalization, and practical implementation using modern DBMS technologies. The course prepares students to design, implement, and manage database systems for real-world applications.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Design database schemas using ER modeling and normalization
CO2	Implement and query databases using SQL and PL/SQL
CO3	Apply transaction management and concurrency control
CO4	Design efficient storage structures and indexing schemes

Unit	Content	Credit	Weightage

I	**Introduction to Database Systems**
	
	• **1.1 Database System Concepts**
	  ○ Data, information, and knowledge
	  ○ File systems vs database systems
	  ○ Three-schema architecture (internal, conceptual, external)
	  ○ Data independence (logical and physical)
	• **1.2 Database System Architecture**
	  ○ DBMS components: Query processor, storage manager
	  ○ Database users and administrators
	  ○ Database languages: DDL, DML, DCL
	• **1.3 Database Applications**
	  ○ Traditional applications (banking, airlines, universities)
	  ○ Modern applications (e-commerce, social media, IoT)
	  ○ Emerging trends (big data, cloud databases)
		1	25%

II	**Data Models and Database Design**
	
	• **2.1 Entity-Relationship Model**
	  ○ Entities, attributes, relationships
	  ○ ER diagram notation (Chen notation, Crow's foot)
	  ○ Cardinality constraints (1:1, 1:N, M:N)
	  ○ Weak entities, composite attributes, multivalued attributes
	• **2.2 Enhanced ER Modelling**
		1	25%

	○ Specialization and generalization ○ Aggregation ○ Inheritance in EER • **2.3 Relational Model** ○ Relations, tuples, attributes, domains ○ Keys: Super, candidate, primary, foreign ○ Relational algebra operations ▪ Basic: select, project, union, set difference, Cartesian product ▪ Additional: rename, intersection, natural join, division ○ Relational calculus (tuple and domain)		
III	**Structured Query Language (SQL)** • **3.1 SQL Fundamentals** ○ Data types, schema definition ○ Basic queries: SELECT, FROM, WHERE ○ Aggregate functions, GROUP BY, HAVING ○ Set operations: UNION, INTERSECT, EXCEPT • **3.2 Advanced SQL** ○ Nested subqueries (correlated and non-correlated) ○ JOIN operations: INNER, LEFT, RIGHT, FULL OUTER ○ Views: creation, updating, materialized views ○ Integrity constraints: NOT NULL, UNIQUE, CHECK, DEFAULT • **3.3 SQL Programming** ○ PL/SQL or T-SQL basics ○ Stored procedures, functions, triggers ○ Cursors, exception handling ○ Embedded SQL, dynamic SQL	1	25%
IV	**Database Design Theory** • **4.1 Functional Dependencies** ○ Definition and properties ○ Armstrong's axioms ○ Closure of attribute sets ○ Canonical cover • **4.2 Normalization** ○ First Normal Form (1NF) ○ Second Normal Form (2NF) ○ Third Normal Form (3NF) ○ Boyce-Codd Normal Form (BCNF) ○ Higher normal forms (4NF, 5NF) overview	1	25%

	• **4.3 Decomposition Algorithms** ○ Lossless join decomposition ○ Dependency preserving decomposition ○ Synthesis algorithm for 3NF ○ Decomposition algorithm for BCNF		

Textbooks:

- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe
- "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke

Reference Books:

- "SQL and Relational Theory: How to Write Accurate SQL Code" by C.J. Date
- "Transaction Processing: Concepts and Techniques" by Jim Gray and Andreas Reuter
- "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" by Pramod J. Sadalage and Martin Fowler
- "Data Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems" by Martin Kleppmann

Online Resources:

- Stanford Online: Databases course by Jennifer Widom
- Coursera: Database series by University of Michigan
- NPTEL: Database Management Systems courses
- PostgreSQL Documentation
- MySQL Reference Manual
- MongoDB University (free courses)
- db-fiddle.com (online SQL playground)

Laboratory Experiments

- **Lab 1:** Installation and configuration of DBMS (MySQL/PostgreSQL)
- **Lab 2:** Basic SQL – DDL, DML commands
- **Lab 3:** SQL queries with aggregate functions and grouping
- **Lab 4:** Advanced SQL – Subqueries, joins, views
- **Lab 5:** PL/SQL programming – Stored procedures and functions
- **Lab 6:** Triggers and cursors implementation
- **Lab 7:** ER modeling and conversion to relational schema
- **Lab 8:** Normalization exercises
- **Lab 9:** Index creation and performance analysis
- **Lab 10:** Transaction management and concurrency control
- **Lab 11:** NoSQL database basics (MongoDB)

SEMESTER -IV

COURSE CODE: BTCSE401
COURSE NAME: OPERATING SYSTEM

Course Objective

This course provides a comprehensive understanding of operating system concepts, design principles, and implementation techniques. It covers process management, memory management, storage systems, and system security, focusing on practical applications and modern OS design for computer science engineering.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Analyze and implement process management and schedulingalgorithms
CO2	Design memory management schemes including paging and virtual memory
CO3	Implement file systems and storage management techniques
CO4	Apply security and protection mechanisms in operating systems

Unit	Content	Credit	Weightage
I	Introduction and Process Management 1.1 Introduction to Operating Systems 1.2 Process Management 1.3 CPU Scheduling 1.4 Process Synchronization	1	25%
II	Memory Management 2.1 Memory Management Basics 2.2 Paging 2.3 Segmentation 2.4 Virtual Memory	1	25%
III	Storage Management and File Systems 3.1 Mass-Storage Structure 3.2 File System Interface 3.3 File System Implementation, 3.4 I/O Systems	1	25%

IV	**System Security**
	
	• **Security Problems**
	  ◦ Program threats: Trojan horse, Trap door, Logic bomb, Stack and buffer overflow
	  ◦ System and network threats: Worms, Port scanning, DoS
	• **Security Measures**
	  ◦ Authentication: Passwords, Physical identification, Biometrics
	  ◦ One-time passwords
	  ◦ Cryptography: Symmetric and Asymmetric encryption
	  ◦ Digital signatures, Certificates
	  ◦ Firewalls, Intrusion detection systems
	
	4.3 Virtual Machines
	
	• **Virtual Machine Concepts**
	  ◦ Benefits of virtualization
	  ◦ Implementation methods
	  ◦ Examples: VMware, Java Virtual Machine
	• **Containerization**
	  ◦ Docker architecture
	  ◦ Containers vs Virtual Machines
	  ◦ Container orchestration basics (Kubernetes overview)
	
	4.4 Distributed Systems
	
	• **Distributed System Structure**
	  ◦ Types of distributed OS
	  ◦ Network structure, Communication protocols
	• **Distributed File Systems**
	  ◦ NFS architecture
	  ◦ AFS design
	• **Distributed Coordination**
	  ◦ Event ordering, Mutual exclusion
	  ◦ Atomicity, Concurrency control
	  ◦ Deadlock handling in distributed systems
	
	4.5 Case Studies and Modern OS Features
	
	Linux Case Study, Windows Case Study
		1	25%

Textbooks:

- "Operating System Concepts" by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne (10th edition)
- "Modern Operating Systems" by Andrew S. Tanenbaum (4th edition)

Reference Books:

- "Operating Systems: Three Easy Pieces" by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau (Free online)
- "The Design of the UNIX Operating System" by Maurice J. Bach
- "Windows Internals" by Mark Russinovich, David Solomon, and Alex Ionescu

Online Resources:

- NPTEL: Operating Systems course by Prof. Chester Rebeiro (IIT Madras)
- MIT Open Course Ware: Operating System Engineering
- Linux Kernel Documentation: kernel.org/doc/html/latest/
- OS Dev Wiki: osdev.org
- Geeks for Geeks OS Section

COURSE CODE: BTCSE402
COURSE NAME: THEORY OF COMPUTATION
Course Objective

This course introduces the mathematical foundations of computation, covering automata theory, formal languages, computability, and complexity theory. It provides the theoretical basis for understanding what can be computed, how efficiently it can be computed, and the fundamental limits of computation.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Design and analyze finite automata and regular expressions
CO2	Construct context-free grammars and pushdown automata for formal languages
CO3	Prove language properties using pumping lemmas and closure properties
CO4	Design Turing machines and understand computability limits

Unit	Content	Credit	Weightage
I	Automata Theory and Regular Languages 1.1 Introduction to Theory of Computation 1.2 Finite Automata 1.3 Regular Expressions and Languages 1.4 Regular Language Properties	1	25%
II	Context-Free Languages and Pushdown Automata 2.1 Context-Free Grammars 2.2 Pushdown Automata 2.3 Properties of Context-Free Languages 2.4 Applications of CFGs and PDAs	1	25%

III	Turing Machines and Computability Theory 3.1 Turing Machines 3.2 Decidability and Recursive Languages 3.3 Reducibility 3.4 Advanced Computability Topics	1	25%
IV	Complexity Theory 4.1 Time and Space Complexity 4.2 NP-Completeness 4.3 Advanced Complexity Classes 4.4 Current Frontiers and Applications	1	25%

Textbooks:

- "Introduction to the Theory of Computation" by Michael Sipser (3rd edition)
- "Automata Theory, Languages, and Computation" by John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman (3rd edition)

Reference Books:

- "Elements of the Theory of Computation" by Harry R. Lewis and Christos H. Papadimitriou
- "Computational Complexity: A Modern Approach" by Sanjeev Arora and Boaz Barak
- "Computability, Complexity, and Languages" by Davis, Sigal, and Weyuker

Online Resources:

- NPTEL: Theory of Computation by Prof. Somenath Biswas (IIT Kanpur)
- MIT Open Course Ware: Automata, Computability, and Complexity
- Stanford Online: Automata and Complexity Theory
- Pumping Lemma Visualizations: Multiple online tools
- Turing Machine Simulators: Online and downloadable

COURSE CODE: BTCSE403
COURSE NAME: MICROPROCESSORS AND MICROCONTROLLERS
Course Objective

This course introduces the architecture, programming, and interfacing of microprocessors and microcontrollers with emphasis on embedded systems development. It covers 8086 microprocessor and 8051 microcontroller architectures, assembly programming, interfacing techniques, and embedded system design principles for computer science applications.

Course Outcomes (COs)

Upon successful completion, students will be able to:

CO1	Analyze and program 8086 microprocessor using assembly language
CO2	Design interfacing circuits for 8086 with memory and peripherals
CO3	Program 8051 microcontroller using both assembly and embedded
CO4	Interface sensors, actuators, and displays with 8051 microcontroller

Unit	Content	Credit	Weightage
I	8086 Microprocessor Architecture and Programming 1.1 Introduction to Microprocessors 1.2 8086 Microprocessor Architecture 1.3 8086 Assembly Language Programming 1.4 8086 Interrupts and DMA	1	25%
II	8086 Interfacing and System Design 2.1 Memory Interfacing 2.2 I/O Interfacing 2.3 Interfacing with Peripherals 2.4 System Design Examples	1	25%
III	8051 Microcontroller Architecture and Programming 3.1 Introduction to Microcontrollers 3.2 8051 Architecture 3.3 8051 Instruction Set and Programming 3.4 C Programming for 8051	1	25%
IV	8051 Interfacing and Embedded Systems 4.1 8051 Interfacing Techniques 4.2 Interrupt System 4.3 Advanced Interfacing 4.4 Embedded Systems Design	1	25%

Textbooks:

- "The 8086 Microprocessor: Programming and Interfacing the PC" by Kenneth J. Ayala
- "The 8051 Microcontroller and Embedded Systems" by Muhammad Ali Mazidi, Janice Gillispie Mazidi, and Rolin D. McKinlay

Reference Books:

- "Advanced Microprocessors and Peripherals" by A.K. Ray and K.M. Bhurchandi
- "Microprocessor Architecture, Programming, and Applications with the 8085" by Ramesh S. Gaonkar
- "Embedded C Programming and the Microchip PIC" by Richard H. Barnett
- "Making Embedded Systems: Design Patterns for Great Software" by Elecia White

Online Resources:

- NPTEL: Microprocessors and Microcontrollers courses
- 8051.com - Tutorials and resources
- Keil Development Tools: keil.com
- Proteus Simulation Software: labcenter.com
- Arduino Platform: arduino.cc (for modern microcontroller concepts)

Laboratory Experiments

8086 Experiments:

1. **Lab 1:** Assembly language programs for arithmetic operations
2. **Lab 2:** String manipulation and array processing
3. **Lab 3:** Interfacing with 8255 PPI
4. **Lab 4:** ADC and DAC interfacing
5. **Lab 5:** Stepper motor control

8051 Experiments:

6. **Lab 6:** 8051 programming in assembly and C
7. **Lab 7:** LED patterns and seven-segment displays
8. **Lab 8:** LCD interfacing and programming
9. **Lab 9:** Keyboard interfacing
10. **Lab 10:** Timer programming and PWM generation
11. **Lab 11:** Serial communication with PC
12. **Lab 12:** Temperature monitoring system
13. **Lab 13:** DC motor speed control

COURSE CODE: BTCSE404
COURSE NAME: ADVANCED DATA STRUCTURE
Course Objective

This course explores advanced data structures beyond basic lists, stacks, and queues. It covers efficient data organization techniques, probabilistic structures, geometric structures, and external memory structures. Emphasis is on theoretical analysis, practical implementation, and applications in modern computing domains like databases, networks, and computational geometry.

CO1	Design and implement balanced tree structures for efficient search operations
CO2	Apply probabilistic structures for approximate query processing in large datasets
CO3	Implement geometric structures for spatial data processing and queries
CO4	Design concurrent and persistent data structures for multi-threaded applications

Unit	Content	Credit	Weightage
I	Advanced Tree Structures 1.1 Balanced Search Trees 1.2 Advanced Tree Variants 1.3 Trie Structures 1.4 Spatial Trees	1	25%
II	Hashing and Probabilistic Data Structures 2.1 Advanced Hashing Techniques 2.2 Bloom Filters and Variants 2.3 Sketching Data Structures 2.4 Streaming Algorithms Structures	1	25%
III	Geometric and Graph-Based Structures 3.1 Geometric Data Structures 3.2 Priority Queue Variants 3.3 Graph-Based Structures	1	25%

	3.4 Succinct Data Structures		
IV	External Memory and Concurrent Structures 4.1 External Memory Structures 4.2 Concurrent Data Structures 4.3 Persistent Data Structures 4.4 Specialized Structures and Applications	1	25%

Textbooks:

- "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein (Chapters on advanced structures)
- "The Art of Computer Programming, Volume 3: Sorting and Searching" by Donald E. Knuth
- "Data Structures and Algorithms in Python" by Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser

Reference Books:

- "Handbook of Data Structures and Applications" edited by Dinesh P. Mehta and Sartaj Sahni
- "Algorithms on Strings, Trees, and Sequences" by Dan Gusfield
- "Computational Geometry: Algorithms and Applications" by Mark de Berg et al.
- The Design and Analysis of Concurrent Data Structures" by Mark Moir and Nir Shavit

Online Resources:

- MIT Open Course Ware: Advanced Data Structures
- Stanford CS166: Data Structures (online materials)
- Visual Algo: Advanced data structure visualizations
- CP-Algorithms: Advanced topics with implementations
- Research Papers: Key papers on modern data structures

Laboratory Experiments

1. **Lab 1:** AVL and Red-Black tree implementation with visualization
2. **Lab 2:** B-Tree implementation for disk-based storage simulation
3. **Lab 3:** Bloom filter variants and performance comparison
4. **Lab 4:** Count-Min Sketch for stream frequency estimation
5. **Lab 5:** kd-Tree for nearest neighbor search in 2D
6. **Lab 6:** Fibonacci heap and Dijkstra's algorithm optimization
7. **Lab 7:** Concurrent linked list and queue implementation
8. **Lab 8:** Persistent data structure implementation
9. **Lab 9:** Suffix array construction and pattern matching
10. **Lab 10:** Final project - Real-world application using advanced structures

COURSE CODE: BTCSE405
COURSE NAME: INTRODUCTION TO IOT
Course Objective

This course introduces the fundamentals of Internet of Things (IoT) systems, covering architecture, protocols, sensing/actuation, data processing, and security. Emphasis is placed on hands-on development of IoT applications using popular platforms, understanding end-to-end IoT systems, and exploring real-world applications in smart cities, healthcare, and industry.

CO1	Design IoT system architectures for different application domains
CO2	Select appropriate communication protocols based on application requirements
CO3	Develop IoT applications using popular hardware platforms and cloud services
CO4	Implement security measures for IoT devices and communications

Unit	Content	Credit	Weightage
I	IoT Fundamentals and Architecture 1.1 Introduction to IoT 1.2 IoT Architecture and Layers 1.3 IoT Hardware Components 1.4 IoT Software Stack	1	25%
II	IoT Connectivity and Communication Protocols 2.1 Short-Range Wireless Protocols 2.2 LPWAN (Low Power Wide Area Networks) 2.3 IoT Network Architecture 2.4 IoT Application Layer Protocols	1	25%
III	IoT Data and Platform Services 3.1 IoT Data Processing and Analytics 3.2 IoT Cloud Platforms 3.3 IoT Application Development 3.4 IoT and Emerging Technologies	1	25%
IV	IoT Security, Privacy, and Applications 4.1 IoT Security Fundamentals 4.2 IoT Attacks and Countermeasures 4.3 IoT Privacy and Ethics 4.4 IoT Applications and Case Studies	1	25%

Textbooks:

- "Internet of Things: Principles and Paradigms" by Rajkumar Buyya and Amir Vahid Dastjerdi
- "Getting Started with the Internet of Things" by Cuno Pfister
- "Building the Internet of Things with ESP32 and NodeJS" by Cătălin Mariș

Reference Books:

- "IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things" by David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, Jerome Henry
- "The Internet of Things: Enabling Technologies, Platforms, and Use Cases" by Pethuru Raj and Anupama C. Raman
- "Practical IoT Hacking" by Fotios Chantzis, Ioannis Stais, Paulino Calderon, and Beau Woods

Online Resources:

- IoT Alliance Guidelines and Whitepapers
- AWS IoT Tutorials and Documentation
- Azure IoT Learning Paths
- Arduino Project Hub (IoT projects)
- Hackster.io (IoT project community)
- IoT For All (Online publication)
- IEEE IoT Journal

Laboratory Experiments

- Lab 1: Setting up IoT development environment
- Lab 2: Sensor interfacing and data acquisition (DHT22, MPU6050)
- Lab 3: Wi-Fi connectivity and basic HTTP communication
- Lab 4: MQTT implementation with local broker (Mosquitto)
- Lab 5: Cloud integration (AWS IoT/Azure IoT)
- Lab 6: BLE communication and beacon implementation
- Lab 7: Edge computing with Raspberry Pi (data preprocessing)
- Lab 8: Building IoT dashboard with Node-RED
- Lab 9: Implementing security features (TLS, authentication)
- Lab 10: Complete IoT project - Smart home/Environmental monitoring

SEMESTER-V

COURSE CODE: BTCSE501
COURSE NAME: COMPUTER NETWORK
Course Objective

- Understand fundamental networking concepts, models, and layered architectures.
- Analyze data link layer protocols, error control, and media access mechanisms.
- Design IP addressing schemes and explain routing protocols.
- Evaluate transport layer protocols (TCP/UDP) and congestion control mechanisms.
- Implement basic network applications and analyze network performance.

Course Outcomes:

CO1	Explain network architectures, physical layer technologies, and transmission fundamentals
CO2	Apply error detection/correction techniques and data link layer protocols
CO3	Design subnetting strategies and compare routing algorithms.
CO4	Analyze TCP/UDP operations and implement client-server applications.

Unit	Content	Credit	Weightage
I	**Fundamentals & Physical Layer** • **Introduction to Networks:** Uses, types (LAN, MAN, WAN), network topologies, network models (OSI, TCP/IP). • **Physical Layer:** Transmission media (guided/unguided), digital/analog transmission, multiplexing (FDM, TDM), switching (circuit, packet, message), bandwidth, throughput, latency. • **Data Encoding & Modulation:** NRZ, Manchester, PCM, ASK, FSK, PSK. • **Introduction to Network Hardware:** Hubs, repeaters, cables, connectors.	1	25%
II	**Data Link Layer & MAC Sublayer** • **Data Link Layer Design Issues:** Framing, error control, flow control. • **Error Detection & Correction:** Parity, checksum, CRC, Hamming codes. • **Data Link Protocols:** Stop-and-Wait, Sliding Window, Go-Back-N, Selective Repeat. • **MAC Sublayer:** Channel allocation, multiple access protocols – ALOHA, CSMA, CSMA/CD, CSMA/CA. • **LAN Technologies:** Ethernet (802.3), Wireless LANs (802.11), VLANs, PPP.	1	25%

	• **Switching:** Bridges vs. switches, MAC learning, spanning tree protocol.		
III	**Network Layer & Routing** • **Network Layer Functions:** Logical addressing, routing, fragmentation. • **IP Addressing:** IPv4, subnetting, CIDR, IPv6 basics. • **Routing Algorithms:** Static vs. dynamic routing, distance vector (RIP), link state (OSPF), path vector (BGP). • **IP Protocol:** IP datagram format, ARP, ICMP, DHCP. • **Network Layer Devices:** Routers, routing tables. • **Introduction to QoS and Multicast.**	1	25%
IV	**Transport Layer & Application Layer** • **Transport Layer:** Role and services, connectionless vs. connection-oriented. • **TCP & UDP:** Segment format, TCP connection management (3-way handshake), flow control, congestion control (slow start, AIMD), UDP features. • **Application Layer Protocols:** DNS, HTTP/HTTPS, FTP, SMTP, POP/IMAP. • **Introduction to Network Security:** Basic concepts of firewalls, TLS/SSL, VPNs. • **Socket Programming Basics.**	1	25%

Textbooks:

- "Computer Networking: A Top-Down Approach"
  Kurose & Ross (7th Edition) – Excellent for application-first learning.
- "Computer Networks"
  Andrew S. Tanenbaum & David J. Wetherall (5th Edition) – Strong on theory and layers.
- "Data Communications and Networking"
  Behrouz A. Forouzan (5th Edition) – Clear explanations with good visuals.

Reference Books:

- "TCP/IP Illustrated, Vol. 1" *W. Richard Stevens* – In-depth protocol details.
- "Internetworking with TCP/IP, Vol. 1" *Douglas E. Comer* – Classic TCP/IP reference.
- "Network Programmability and Automation" *Jason Edelman et al.* – For modern network automation topics.

Online Resources:

- Cisco Networking Academy courses
- Geeks for Geeks Computer Networks section
- Wireshark documentation and sample captures

Laboratory Experiments
Simulation & Programming (Using Cisco Packet Tracer / NS3)

- **Experiment 1:** Study of network devices and cables.

- **Experiment 2:** Basic network configuration and connectivity testing.
- **Experiment 3:** Implementation of framing methods (Bit/Byte stuffing).
- **Experiment 4:** Error detection using CRC and Hamming codes.
- **Experiment 5:** Stop-and-Wait and Sliding Window protocol simulation.
- **Experiment 6:** LAN setup and VLAN configuration.
- **Experiment 7:** Subnetting practice and IP address allocation.
- **Experiment 8:** Static and dynamic routing (RIP/OSPF) configuration.
- **Experiment 9:** TCP and UDP client-server programs (Socket programming).
- **Experiment 10:** DNS and HTTP packet analysis.

COURSE CODE: BTCSE502
COURSE NAME: DESIGN OF ALGORITHMS AND ANALYSIS
Course Objective

- Analyze algorithms using asymptotic notations and recurrence relations.
- Compare various algorithm design paradigms and their applications.
- Design efficient algorithms using appropriate design techniques.
- Evaluate computational complexity of problems.
- Implement algorithms and analyze their performance empirically.

Course Outcomes:

CO1	Analyze time and space complexity of algorithms using asymptotic notations.
CO2	Apply divide-and-conquer and greedy techniques to solve computational problems
CO3	Design dynamic programming solutions for optimization problems.
CO4	Implement backtracking and branch-and-bound algorithms

Unit	Content	Credit	Weightage
I	Foundations and Analysis of Algorithms • Introduction: Algorithm definition, characteristics, algorithm vs. program • Algorithm Design Techniques: Brute force, divide and conquer, decrease and conquer • Algorithm Analysis • Mathematical Foundations: Growth of functions, summations, solving recurrences • Sorting Algorithms Analysis: Comparison of Bubble, Selection, Insertion sort	1	25%
II	Divide and Conquer & Greedy Methods Divide and Conquer Strategy, Greedy Algorithms	1	25%
III	Dynamic Programming & Backtracking Dynamic Programming, Backtracking	1	25%

IV	Advanced Topics & Complexity Theory Branch and Bound, String Matching Algorithms, Complexity Theory, Approximation Algorithms, Introduction to Randomized Algorithms	1	25%

Textbooks:

- "Introduction to Algorithms" *Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein* (4th Edition) *(CLRS)* – The standard reference, comprehensive coverage.
- "Algorithms" *Robert Sedgewick & Kevin Wayne* (4th Edition) *Excellent for implementations and practical approach*.
- "The Algorithm Design Manual" *Steven S. Skiena* (3rd Edition) *Great for problem-solving patterns and interview preparation*.

Reference Books:

- "Algorithm Design" *Jon Kleinberg & Éva Tardos Strong on design techniques and applications*.
- "Fundamentals of Computer Algorithms" *Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran Good for theoretical foundations*.
- "Data Structures and Algorithms in Python/Java/C++" *Michael T. Goodrich, Roberto Tamassia Excellent for implementation-focused learning*.

Online Resources:

- MIT Open Course Ware: 6.006 Introduction to Algorithms
- Geeks for Geeks: DSA section with implementations
- Leet Code/Coding Platforms: For practice problems
- Visu Algo: Algorithm visualizations
- NPTEL: "Design and Analysis of Algorithms" by Prof. Madhavan Mukund

COURSE CODE: BTCSE503

COURSE NAME: SOFTWARE ENGINEERING

Course Objective

- Understand software development life cycle models and their applicability.
- Apply requirements engineering techniques and develop SRS documents.
- Design software architectures using UML diagrams and design principles.
- Implement software testing strategies and quality assurance practices.
- Manage software projects using estimation, scheduling, and risk management techniques.
- Analyze emerging trends in software engineering and agile methodologies.

Course Outcomes:

CO1	Compare various SDLC models and select appropriate models for given scenarios.
CO2	Develop Software Requirements Specification (SRS) using requirements elicitation techniques
CO3	Design software systems using UML diagrams and architectural patterns
CO4	Apply testing techniques and develop test cases for software validation

Unit	Content	Credit	Weightage
I	**Software Process & Requirements Engineering** • **Introduction to Software Engineering:** Characteristics, myths, challenges • **Software Process Models:** Waterfall, Prototyping, Incremental, Spiral, RAD • **Agile Development:** Scrum, XP, Kanban, Lean • **Requirements Engineering**	1	25%

II	**Software Design & UML Modelling**
	• **Design Concepts:** Abstraction, architecture, patterns, modularity
	• **Architectural Design:** MVC, Layered, Client-Server, Microservices
	• **UML Diagrams:**
	    o Structural: Class, Object, Component, Deployment diagrams
	    o Behavioral: Use Case, Sequence, Activity, State Machine diagrams
	• **Object-Oriented Design:** Principles (SOLID), design patterns (Factory, Observer, Singleton)
	• **User Interface Design:** Principles, evaluation methods
		1	25%

III	**Software Testing & Quality Assurance**
	• **Testing Fundamentals:** Verification vs. Validation, test levels (unit, integration, system, acceptance)
	• **Testing Techniques:**
	    o White-box testing (Statement, Branch, Path coverage)
	    o Black-box testing (Equivalence partitioning, Boundary value analysis)
	    o Gray-box testing
	• **Test Management:** Test plans, cases, scripts, reports
	• **Software Quality:** ISO 9126, CMMI, Six Sigma
	• **Quality Assurance vs. Quality Control**
		1	25%

IV	**Project Management & Advanced Topics** • **Project Planning:** Estimation techniques (COCOMO, FP, Use Case Points) • **Scheduling:** Gantt charts, PERT, CPM, Critical path method • **Risk Management:** Identification, analysis, mitigation, monitoring • **Configuration Management:** Version control (Git), build management • **Software Maintenance:** Types (corrective, adaptive, perfective, preventive), reverse engineering • **Emerging Trends:** DevOps, CI/CD, Cloud-native development, AI in SE • **Ethics in Software Engineering:** Professional responsibilities	1	25%

Textbooks:

- "Software Engineering: A Practitioner's Approach" *Roger S. Pressman & Bruce Maxim* (9th Edition) *Comprehensive coverage, industry standard*
- "Software Engineering" *Ian Sommerville* (10th Edition) *Excellent for academic*

perspective, clear explanations

- "Head First Software Development" *Dan Pilone & Russ Miles Great for beginners, visual learning approach*

Reference Books:

- "The Mythical Man-Month"*Frederick P. Brooks Jr.Classic essays on software engineering*
- "Clean Code: A Handbook of Agile Software Craftsmanship" *Robert C. Martin Essential for coding practices*
- "Design Patterns: Elements of Reusable Object-Oriented Software" *Gamma, Helm, Johnson, Vlissides* (Gang of Four) *Patterns reference*

Online Resources:

- IEEE Software Engineering Standards
- Martin Fowler's Blog (Refactoring, patterns)
- Agile Alliance Resources
- Coursera/Udacity: Software Engineering courses
- NPTEL: "Software Engineering" by Prof. Rajib Mall

Laboratory Experiments

- Session 1: Study of various SDLC models with case studies
- Session 2: Developing SRS document for a sample system
- Session 3: Creating Use Case diagrams and descriptions
- Session 4: Designing Class diagrams and Sequence diagrams
- Session 5: Developing Activity diagrams and State diagrams
- Session 6: Version control with Git (basic commands, branching)

  Session 7: Developing test cases using equivalence partitioning
  Session 8: White-box testing and calculating cyclomatic complexity
  Session 9: Developing test plans and test reports
  Session 10: Code review and refactoring exercises

COURSE CODE: BTCSE504
COURSE NAME: COMPUTER GRAPHICS
Course Objective

- Understand fundamental graphics hardware, display technologies, and graphics pipeline.
- Apply mathematical transformations and algorithms for 2D/3D graphics.
- Implement scan conversion algorithms and clipping techniques.
- Design 3D viewing pipelines, projections, and visible surface detection methods.
- Develop interactive graphics applications using OpenGL/WebGL.
- Analyze illumination models, color models, and rendering techniques.

Course Outcomes:

CO1	Explain graphics hardware, display technologies, and graphics pipeline architecture.
CO2	Implement 2D transformation algorithms and viewing

	pipelines
CO3	Apply 3D transformation, projection, and visible surface detection algorithms
CO4	Develop graphics programs using OpenGL for interactive applications

Unit	Content	Credit	Weightage
I	**Introduction & Graphics Fundamentals** • **Introduction to Computer Graphics:** Applications (CAD, Games, Visualization, VR/AR) • **Graphics Systems:** Video display devices (CRT, LCD, LED, Plasma), raster-scan systems • **Graphics Hardware:** Graphics cards, GPU architecture, frame buffers, color depth • **Graphics Pipeline:** Overview of rendering pipeline (fixed vs. programmable) • **Input Devices:** Keyboard, mouse, touch, light pen, data glove • **Basic Drawing Primitives:** Points, lines, polygons • **Scan Conversion Algorithms**	1	25%
II	**2D Transformations & Viewing** **2D Geometric Transformations Windowing and Clipping Raster Operations**	1	25%
III	**3D Graphics & Projections** **3D Concepts, 3D Viewing Pipeline, Projections, Visible Surface Detection**	1	25%
IV	**Advanced Topics & Implementation** • **Color Models , Illumination Models, Rendering Techniques, Curves and Surfaces, Computer Animation**	1	25%

Textbooks:

- "Computer Graphics with OpenGL" *Donald D. Hearn, M. Pauline Baker, Warren Carithers* (4th Edition) *Excellent for OpenGL integration with theory*
- "Interactive Computer Graphics: A Top-Down Approach with OpenGL" *Edward Angel & Dave Shreiner* (7th Edition) *Great for beginner-friendly OpenGL programming*
- "Computer Graphics: Principles and Practice" *John F. Hughes, Andries van Dam, James D. Foley* (3rd Edition) *The "bible" of computer graphics, comprehensive*

Reference Books:

- "Fundamentals of Computer Graphics" *Steve Marschner & Peter Shirley* (5th Edition) *Modern approach with excellent mathematical foundation*
- "OpenGL Programming Guide" (The Red Book) *Dave Shreiner, et al.* (9th Edition) *Official OpenGL guide*

- "Real-Time Rendering" *Tomas Akenine-Möller, Eric Haines, Naty Hoffman* (4th Edition) *Industry standard for game graphics*

Online Resources:

- **LearnOpenGL.com** (Excellent tutorials)
- **OpenGL Documentation** (Khronos Group)
- **Scratchapixel** (Free computer graphics education)
- **ShaderToy** (For shader programming)
- **NPTEL:** "Computer Graphics" by Prof. Sukhendu Das

COURSE CODE: BTCSE505
COURSE NAME: CLOUD COMPUTING
Course Objective

- Understand cloud computing concepts, models, and service architectures.
- Analyze virtualization technologies and cloud deployment models.
- Design and deploy applications using cloud platforms (AWS/Azure/GCP).
- Implement storage, compute, and networking solutions in cloud environments.
- Evaluate cloud security, compliance, and cost management strategies.
- Compare emerging cloud technologies and migration approaches.

Course Outcomes:

CO1	Explain cloud computing characteristics, service models, and deployment models.
CO2	Analyze virtualization techniques and containerization technologies
CO3	Deploy and manage applications using major cloud platforms
CO4	Implement cloud storage, networking, and compute services

Unit	Content	Credit	Weightage
I	Cloud Foundations & Virtualization, Introduction to Cloud Computing, Cloud Service Models, Cloud Deployment Models, Virtualization Fundamentals	1	25%
II	Cloud Architecture & Core Services, Cloud Reference Architecture, Compute Services, Storage Services, Networking in Cloud, Database Services	1	25%
III	Cloud Management & Security • Cloud Management & Monitoring, Cloud Security, Compliance & Governance, Cost Management	1	25%
IV	Advanced Cloud Services & Trends	1	25%

	• Cloud-Native Development, Big Data & Analytics in Cloud, Multi-cloud & Hybrid Cloud, Emerging Trends		

Textbooks:

- "Cloud Computing: Concepts, Technology & Architecture" *Thomas Erl, Ricardo Puttini, Zaigham Mahmood Comprehensive coverage of cloud patterns and best practices*
- "Cloud Computing: A Practical Approach" *Anthony T. Velte, Toby J. Velte, Robert Elsenpeter Excellent for hands-on learning and implementation*
- "Architecting the Cloud: Design Decisions for Cloud Computing Service Models" *Michael J. Kavis Great for architectural decision-making*

Reference Books:

- "The Cloud Adoption Playbook" *Moe Abdula, Ingo Averdunk, et al. Practical guide for enterprise cloud adoption*
- "Site Reliability Engineering: How Google Runs Production Systems" *Betsy Beyer, et al. SRE principles in cloud context*
- "Cloud Native Patterns: Designing Change-tolerant Software" *Cornelia Davis Modern cloud-native application design*

Online Resources:

- AWS/Azure/GCP Documentation & Free Tier
- Cloud Academy, A Cloud Guru (Video courses)
- NPTEL: "Cloud Computing" by Prof. Soumya Kanti Ghosh
- Cloud Computing Specialization (Coursera - UIUC)
- CNCF (Cloud Native Computing Foundation) Resources

SEMESTER-VI

COURSE CODE: BTCSE601
COURSE NAME: COMPILER DESIGN
Course Objective

- Understand the structure and phases of compiler design.
- Design lexical analyzers and parsers for programming languages.
- Implement syntax-directed translation and intermediate code generation.
- Optimize code using various optimization techniques.
- Generate efficient target code for different architectures.
- Analyze compiler construction tools and modern compiler trends

Course Outcomes:

CO1	Explain compiler phases, language processing systems, and compiler construction tools.
CO2	Design and implement lexical analyzers using regular expressions and finite automata
CO3	Construct parsers using top-down and bottom-up parsing techniques
CO4	Generate intermediate code using syntax-directed translation schemes

Unit	Content	Credit	Weightage
I	Introduction & Lexical Analysis • Introduction to Compilers, Lexical Analysis (Scanner), LEX/FLEX Tool	1	25%
II	Syntax Analysis & Parsing • Syntax Analysis (Parser), Top-Down Parsing, Bottom-Up Parsing	1	25%
III	Syntax-Directed Translation & Intermediate Code Generation • Syntax-Directed Translation, Intermediate Code Generation, Symbol Table Management	1	25%
IV	Code Optimization & Code Generation • Code Optimization, Code Generation, Recent Trends	1	25%

Textbooks:

- "Compilers: Principles, Techniques, and Tools" *Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman* (2nd Edition) *The "Dragon Book" – Standard reference for compiler design*
- "Modern Compiler Implementation in C/Java/ML" *Andrew W. Appel* (Tiger Book Series)

Excellent for practical implementation aspects

- "Engineering a Compiler" *Keith D. Cooper & Linda Torczon* (2nd Edition)
Great for engineering perspective and optimization

Reference Books:

- "Compiler Construction: Principles and Practice" *Kenneth C. Louden Good for beginners with clear explanations*
- "Advanced Compiler Design and Implementation" *Steven S. Muchnick Advanced topics and optimization techniques*
- "Crafting a Compiler" *Charles N. Fischer, Ronald K. Cytron, Richard J. LeBlanc Jr. Modern approach with Java implementation*

Online Resources:

- Stanford University Compilers Course (online lectures)
- NPTEL: "Compiler Design" by Prof. Y.N. Srikant
- MIT Open Course Ware: Compiler courses
- GCC and LLVM Documentation
- Compiler Explorer (godbolt.org) – for seeing generated assembl

COURSE CODE: BTCSE602
COURSE NAME: CRYPTOGRAPHY AND NETWORK SECURITY
Course Objective

- Understand fundamental cryptographic principles, algorithms, and protocols.
- Analyze symmetric and asymmetric encryption techniques and their applications.
- Design secure communication protocols for various network environments.
- Implement cryptographic algorithms and security mechanisms.
- Evaluate network security threats, vulnerabilities, and countermeasures.
- Apply authentication, integrity, and confidentiality mechanisms in real-world scenarios.

Course Outcomes:

CO1	Explain basic cryptographic concepts, classical ciphers, and modern encryption principles.
CO2	Implement and analyze symmetric encryption algorithms (DES, AES, etc.).
CO3	Apply asymmetric cryptography (RSA, ECC) and hash functions in security applications
CO4	Design and analyze authentication protocols and digital signature schemes

Unit	Content	Credit	Weightage

I	Foundations of Cryptography • Introduction to Security, Mathematics for Cryptography, Classical Encryption Techniques, Modern Cryptography Principles	1	25%
II	Symmetric Cryptography & Hash Functions • Symmetric Key Algorithms, Cryptographic Hash Functions	1	25%
III	Asymmetric Cryptography & Key Management • Public Key Cryptography, Digital Signatures, Key Management	1	25%
IV	Network Security Applications Authentication Protocols, Email Security, Web Security, Network Security Mechanisms, System Security	1	25%

Textbooks:

- "Cryptography and Network Security: Principles and Practice" *William Stallings* (8th Edition) *Comprehensive coverage with excellent examples and problems*
- "Network Security: Private Communication in a Public World" *Charlie Kaufman, Radia Perlman, Mike Speciner* (2nd Edition) *Great for protocol-level understanding*
- "Applied Cryptography: Protocols, Algorithms, and Source Code in C" *Bruce Schneier* (2nd Edition) *Classic reference with practical implementations*

Reference Books:

- "Introduction to Modern Cryptography" *Jonathan Katz & Yehuda Lindell* (3rd Edition) *Rigorous mathematical treatment*
- "Handbook of Applied Cryptography" *Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone Comprehensive reference with algorithms*
- "Computer Security: Principles and Practice" *William Stallings & Lawrie Brown Broad coverage including system security*

Online Resources:

- Cryptopals Crypto Challenges (Matasano challenges)
- NIST Cryptographic Standards and Guidelines
- NPTEL: "Cryptography and Network Security" by Prof. Debdeep Mukhopadhyay
- Stanford Cryptography I & II (Coursera)
- OWASP Top 10 Security Risks

Practical List:

- **Session 1:** Implementation of classical ciphers (Caesar, Vigenère)
- **Session 2:** Cryptanalysis of substitution ciphers using frequency analysis
- **Session 3:** DES implementation (using libraries) and understanding rounds
- **Session 4:** AES implementation and comparison with DES
- **Session 5:** Block cipher modes of operation implementation
- **Session 6:** Cryptographic hash functions implementation (SHA-256)

- **Session 7:** HMAC implementation and verification
- **Session 8:** RSA algorithm implementation (key generation, encryption, decryption)
- **Session 9:** Diffie-Hellman key exchange simulation
- **Session 10:** Digital signature implementation using RSA

COURSE CODE: BTCSE603
COURSE NAME: ADVANCE COMPUTER ARCHITECTURE
Course Objective

- Analyze performance metrics and quantitative principles of computer architecture.
- Design pipelined processors with hazard detection and resolution mechanisms.
- Evaluate memory hierarchy design, cache organizations, and virtual memory systems.
- Implement instruction-level parallelism techniques including superscalar and VLIW architectures.
- Compare different parallel architectures including multicore, multiprocessor, and GPU architectures.
- Design power-efficient and reliable computing systems.

Course Outcomes:

CO1	Calculate and analyze performance metrics using CPU performance equations.
CO2	Design and analyze pipelined processors with hazard handling mechanisms
CO3	Evaluate memory hierarchy designs including cache organizations and virtual memory
CO4	Implement instruction-level parallelism techniques and analyze their impact

Unit	Content	Credit	Weightage
I	Performance Evaluation & Instruction Set Principles • Introduction to Computer Architecture, Quantitative Principles, Instruction Set Architecture (ISA), Case Studies	1	25%
II	Pipeline & Instruction-Level Parallelism • Pipelining Fundamentals, Hazard Resolution Techniques, Advanced Pipelining	1	25%
III	Memory Hierarchy & I/O Systems • Memory Hierarchy Design, Advanced Cache Optimizations, Virtual Memory, Storage Systems	1	25%
IV	Parallel Architectures & Emerging Trends	1	25%

	• Parallel Processing Architectures, Multicore Architectures, Specialized Architectures, Emerging Trends		

Textbooks:

- "Computer Architecture: A Quantitative Approach" John *L. Hennessy & David A. Patterson* (6th Edition) *The "Bible" of computer architecture – Comprehensive and authoritative*
- "Computer Organization and Design: The Hardware/Software Interface" *David A. Patterson & John L. Hennessy* (ARM or RISC-V Edition) *Excellent for fundamental concepts with practical examples*

Reference Books:

- "Advanced Computer Architecture: Parallelism, Scalability, Programmability" *Kai Hwang Great coverage of parallel architectures*
- "Modern Processor Design: Fundamentals of Superscalar Processors" *John Paul Shen & Mikko H. Lipasti In-depth coverage of modern processor design*
- "Parallel Computer Architecture: A Hardware/Software Approach" *David E. Culler, Jaswinder Pal Singh, Anoop Gupta Comprehensive parallel architecture coverage*

Online Resources:

- MIT Open Course Ware: 6.823 Computer Systems Architecture
- NPTEL: "Computer Architecture" by Prof. Smruti Ranjan Sarangi
- Computer Architecture - Coursera (Princeton University)
- IEEE/ACM Transactions on Computer Architecture
- SPEC Benchmark Suite documentation

COURSE CODE: BTCSE604
COURSE NAME: NATURAL LANGUAGE PROCESSING
Course Objective

- Understand linguistic foundations and computational challenges in NLP.
- Implement text preprocessing, tokenization, and language modeling techniques.
- Design and evaluate classical and neural NLP models for various tasks.
- Apply sequence models, attention mechanisms, and transformers for NLP tasks.
- Analyse NLP applications in real-world scenarios and evaluate model performance.
- Compare different NLP architectures and their suitability for specific applications.

Course Outcomes:

CO1	Explain linguistic concepts and computational challenges in NLP.
CO2	Implement text preprocessing, tokenization, and language modeling techniques
CO3	Design and evaluate classical NLP models for text classification and sequence labeling.
CO4	Implement neural NLP models including RNNs, LSTMs, and attention mechanisms.

Unit	Content	Credit	Weightage
I	NLP Foundations & Text Processing • Introduction to NLP, Linguistic Fundamentals, Text Preprocessing, Statistical Foundations	1	25%
II	Classical NLP & Sequence Models • Text Representation, Text Classification, Sequence Labeling, Syntax and Parsing	1	25%
III	Neural NLP & Attention Models • Neural Network Basics for NLP, Recurrent Neural Networks, Sequence-to-Sequence Models, Attention Mechanisms, Word Embeddings Advanced	1	25%
IV	Transformers & Advanced Applications • Transformer Architecture, Pre-trained Language Models, Advanced NLP Tasks, Current Trends	1	25%

Textbooks:

- "Speech and Language Processing" *Daniel Jurafsky & James H. Martin* (3rd Edition Draft) *The "Bible" of NLP – Comprehensive and authoritative*
- "Natural Language Processing with Python" *Steven Bird, Ewan Klein, & Edward Loper Practical introduction using NLTK*

Reference Books:

- "Foundations of Statistical Natural Language Processing" *Christopher D. Manning & Hinrich Schütze Excellent for statistical foundations*
- "Neural Network Methods for Natural Language Processing" *Yoav Goldberg Great for neural approaches to NLP*
- "Deep Learning for Natural Language Processing" *Palash Goyal, Sumit Pandey, & Karan Jain Practical deep learning approaches*

Online Resources:

- Stanford CS224N: Natural Language Processing with Deep Learning
- NPTEL: "Natural Language Processing" by Prof. Pushpak Bhattacharyya
- Hugging Face Course (Free NLP course with transformers)
- spaCy documentation and tutorials
- Papers with Code (NLP leaderboards and implementations)
- AllenNLP library and tutorials

COURSE CODE: BTCSE605

COURSE NAME: DISTRIBUTED SYSTEMS

Course Objective

- Understand fundamental concepts, architectures, and challenges of distributed systems.
- Design and analyze communication protocols and remote invocation mechanisms.
- Implement distributed algorithms for coordination, consistency, and replication.
- Evaluate distributed file systems, naming services, and peer-to-peer systems.
- Analyze fault tolerance, security, and scalability issues in distributed systems.
- Design modern distributed applications using microservices and cloud-native patterns

Course Outcomes:

CO1	Explain characteristics, architectures, and design challenges of distributed systems.
CO2	Design and implement inter-process communication and remote invocation mechanisms
CO3	Implement distributed algorithms for coordination, consensus, and consistency
CO4	Analyze and design distributed file systems and naming services

Unit	Content	Credit	Weightage
I	Foundations & Communication in Distributed Systems • Introduction to Distributed Systems, System Models, Inter-process Communication, Time and Global State	1	25%
II	Coordination & Distributed Algorithms • Distributed Mutual Exclusion, Election Algorithms, Distributed Transactions & Concurrency Control, Consensus & Agreement, Distributed Coordination Services	1	25%
III	Consistency, Replication & Distributed Storage • Data Consistency Models, Replication, Distributed File Systems, Distributed Naming Services, Peer-to-Peer Systems	1	25%
IV	Fault Tolerance, Security & Modern Distributed Systems • Distributed System Security, Fault Tolerance, Modern Distributed Architectures, Edge & Fog Computing, Distributed Data Processing, Emerging Trends	1	25%

Textbooks:

- "Distributed Systems: Concepts and Design" *George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair* (5th Edition) *Comprehensive coverage with excellent examples*

- "Distributed Systems: Principles and Paradigms" *Andrew S. Tanenbaum & Maarten van Steen* (3rd Edition) *Clear explanations with practical focus*

Reference Books:

- "Designing Data-Intensive Applications" *Martin Kleppmann Modern perspective on distributed data systems*
- "Distributed Algorithms" *Nancy A. Lynch Theoretical foundation of distributed algorithms*
- "Introduction to Reliable and Secure Distributed Programming" *Christian Cachin, Rachid Guerraoui, Luis Rodrigues* (2nd Edition) *Practical approach with implementation examples*

Online Resources:

- MIT 6.824: Distributed Systems (Course materials and labs)
- NPTEL: "Distributed Systems" by Prof. Rajiv Misra
- EdX/Coursera: Cloud Computing and Distributed Systems courses
- Distributed Systems Reading List (University of Cambridge)
- Raft Consensus Algorithm: raft.github.io
- Google Research Papers: GFS, MapReduce, Bigtable, Spanner

Practical List:

Session 1: Socket programming for client-server communication
Session 2: RPC implementation using gRPC/Apache Thrift
Session 3: Lamport timestamp implementation for logical clocks
Session 4: Global snapshot algorithm implementation (Chandy-Lamport)
Session 5: Distributed mutual exclusion algorithm implementation
Session 6: Election algorithm implementation (Bully/Ring)
Session 7: Raft consensus algorithm implementation (basic)
Session 8: ZooKeeper for distributed coordination
Session 9: Simple distributed hash table (DHT) implementation
Session 10: Two-phase commit protocol simulation
Session 11: MapReduce implementation for word count
Session 12: Eventual consistency using gossip protocol

SEMESTER-VII

COURSE CODE: BTCSE701
COURSE NAME: ARTIFICIAL INTELLIGENCE (AI)
Course Objective

- Understand fundamental AI concepts, approaches, and philosophical foundations.
- Design and implement search algorithms for problem-solving.
- Apply knowledge representation and reasoning techniques.
- Implement machine learning algorithms for classification and prediction.
- Design and evaluate neural networks and deep learning models.
- Analyze ethical implications and limitations of AI systems.

Course Outcomes:

CO1	Explain AI concepts, approaches, and intelligent agent architectures.
CO2	Design and implement search algorithms for problem-solving and optimization
CO3	Apply knowledge representation, reasoning, and planning
CO4	Implement machine learning algorithms for classification and regression

Unit	Content	Credit	Weightage
I	Introduction to AI & Problem Solving • Introduction to Artificial Intelligence, Intelligent Agents, Problem Solving by Search, Adversarial Search	1	25%
II	Knowledge Representation & Reasoning • Knowledge Representation, Propositional Logic, First-Order Logic (FOL), Knowledge Engineering, Reasoning Under Uncertainty, Planning	1	25%
III	Machine Learning Fundamentals • Introduction to Machine Learning, Supervised Learning, Unsupervised Learning, Evaluation Metrics	1	25%
IV	Neural Networks & Advanced AI • Neural Networks Fundamentals, Deep Learning, Natural Language Processing (Basics), Reinforcement Learning, Computer Vision (Basics), Ethics in AI, Current Trends	1	25%

Textbooks:

- "Artificial Intelligence: A Modern Approach" *Stuart Russell & Peter Norvig* (4th Edition) *The "Bible" of AI – Comprehensive and authoritative*
- "Pattern Recognition and Machine Learning" *Christopher M. Bishop*

Excellent mathematical foundation for ML

Reference Books:

- "Deep Learning" *Ian Goodfellow, Yoshua Bengio, Aaron Courville Comprehensive deep learning reference*
- "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" *Aurélien Géron* (3rd Edition) *Practical implementation-focused approach*
- "Reinforcement Learning: An Introduction" *Richard S. Sutton & Andrew G. Barto* (2nd Edition) *Standard RL textbook*

Online Resources:

- Stanford CS221: Artificial Intelligence (Course materials)
- NPTEL: "Artificial Intelligence" by Prof. Sudeshna Sarkar
- Andrew Ng's Machine Learning Course (Coursera)
- Fast.ai (Practical deep learning courses)
- Google AI Education
- Papers with Code (Latest AI research with implementations)

COURSE CODE: BTCSE702
COURSE NAME: MACHINE LEARNING
Course Objective

- Understand fundamental ML concepts, learning paradigms, and mathematical foundations.
- Implement and evaluate supervised learning algorithms for regression and classification.
- Design and apply unsupervised learning techniques for clustering and dimensionality reduction.
- Develop and optimize neural networks and deep learning architectures.
- Evaluate model performance using appropriate metrics and validation techniques.
- Analyze ethical considerations and practical challenges in ML deployment.

Course Outcomes:

CO1	Explain ML concepts, learning types, and mathematical foundations.
CO2	Implement and evaluate supervised learning algorithms for regression and classification
CO3	Design and apply unsupervised learning techniques for clustering and dimensionality reduction
CO4	Develop and optimize neural networks and deep learning models

Unit	Content	Credit	Weightage
I	Foundations of Machine Learning • Introduction to Machine Learning, Mathematical Foundations, Data Preprocessing and Feature Engineering, Model Evaluation and Validation	1	25%

II	Supervised Learning • Linear Regression, Logistic Regression, Support Vector Machines (SVM), Decision Trees, Ensemble Methods	1	25%
III	Unsupervised Learning & Dimensionality Reduction • Clustering Algorithms, Dimensionality Reduction, Association Rule Learning	1	25%
IV	Neural Networks & Advanced Topics • Neural Networks Fundamentals, Deep Learning Architectures, Advanced Topics, Model Deployment and Ethics, Current Trends	1	25%

Textbooks:

- "Pattern Recognition and Machine Learning" *Christopher M. Bishop* *Excellent mathematical foundation, comprehensive coverage*
- "The Elements of Statistical Learning" *Trevor Hastie, Robert Tibshirani, Jerome Friedman* (2nd Edition) *Statistical perspective on ML, rigorous treatment*

Reference Books:

- "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" *Aurélien Géron* (3rd Edition) *Practical implementation-focused approach*
- "Machine Learning: A Probabilistic Perspective" *Kevin P. Murphy* *Comprehensive probabilistic approach*
- "Deep Learning" *Ian Goodfellow, Yoshua Bengio, Aaron Courville* *Authoritative deep learning reference*

Online Resources:

- Stanford CS229: Machine Learning (Course materials)
- Andrew Ng's Machine Learning Course (Coursera)
- NPTEL: "Machine Learning" by Prof. Balaraman Ravindran
- Google's Machine Learning Crash Course
- Kaggle Learn (Practical ML tutorials)
- Papers with Code (Latest ML research with implementations)

COURSE CODE: BTCSE703
COURSE NAME: BIG DATA ANALYTICS
Course Objective

- Understand big data characteristics, challenges, and ecosystem components.
- Design and implement distributed storage solutions using Hadoop and HDFS.
- Develop data processing applications using MapReduce and Spark frameworks.
- Analyze big data using machine learning and stream processing techniques.
- Design data pipelines and implement NoSQL databases for big data.
- Evaluate big data architectures and deployment strategies for real-world applications.

Course Outcomes:

CO1	Explain big data characteristics, challenges, and ecosystem components.
CO2	Design and implement distributed storage solutions using Hadoop HDFS
CO3	Develop data processing applications using MapReduce and Apache Spark
CO4	Implement stream processing and machine learning on big data platforms

Unit	Content	Credit	Weightage
I	Big Data Foundations & Hadoop Ecosystem • Introduction to Big Data, Big Data Ecosystem, Hadoop Architecture, Big Data Storage Formats, Data Ingestion Tools	1	25%
II	Data Processing with MapReduce & Spark • MapReduce Programming Model, Apache Spark Fundamentals, Spark Programming	1	25%
III	NoSQL Databases & Stream Processing • NoSQL Databases, Apache HBase, Apache Cassandra, Stream Processing	1	25%
IV	Advanced Analytics & Big Data Applications • Machine Learning on Big Data, Graph Processing, Big Data Analytics, Big Data Security, Big Data DevOps, Real-world Case Studies, Emerging Trends	1	25%

Textbooks:

- "Hadoop: The Definitive Guide" *Tom White* (4th Edition) *Comprehensive Hadoop reference, industry standard*
- "Learning Spark: Lightning-Fast Data Analytics" *Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee* (2nd Edition) *Excellent Spark guide from Databricks creators*

Reference Books:

- "Big Data: Principles and Best Practices of Scalable Realtime Data Systems" *Nathan Marz & James Warren Lambda architecture and real-time systems*
- "Designing Data-Intensive Applications" *Martin Kleppmann Modern perspective on distributed data systems*
- "Spark: The Definitive Guide" *Bill Chambers & Matei Zaharia Comprehensive Spark coverage*

Online Resources:

- Hadoop & Spark Official Documentation
- Cloudera/Hortonworks Tutorials
- Databricks Academy (Free Spark courses)
- Google Cloud Big Data and Machine Learning Fundamentals (Coursera)
- AWS Big Data Specialty Certification Resources
- Apache Project Documentation (Kafka, Flink, Airflow)

Practical List:

Session 1: Hadoop cluster setup (single-node with Docker)
Session 2: HDFS operations and file management
Session 3: MapReduce Word Count and custom implementations
Session 4: Data ingestion with Sqoop and Flume
Session 5: Spark RDD operations and transformations
Session 6: Spark Data Frame operations and SQL queries
Session 7: Spark performance tuning and optimization
Session 8: Spark MLlib for machine learning
Session 9: MongoDB CRUD operations and aggregation
Session 10: HBase operations and data modeling
Session 11: Kafka producers and consumers

COURSE CODE: BTCSE704
COURSE NAME: CYBER SECURITY
Course Objective

- Understand cybersecurity fundamentals, threats, vulnerabilities, and attack vectors.
- Analyze network security mechanisms, cryptography, and secure communication protocols.
- Implement security controls, intrusion detection, and prevention systems.
- Design secure systems, applications, and networks with defense-in-depth strategies.
- Evaluate ethical, legal, and compliance aspects of cybersecurity.
- Respond to security incidents and implement disaster recovery plans.

Course Outcomes:

CO1	Explain cybersecurity concepts, threats, vulnerabilities, and attack methodologies.
CO2	Analyze and implement network security mechanisms and cryptographic protocols

CO3	Design and implement security controls for systems and applications
CO4	Configure and evaluate intrusion detection/prevention systems and firewalls

Unit	Content	Credit	Weightage
I	Cybersecurity Foundations & Cryptography • Introduction to Cybersecurity, Cryptography Fundamentals, Network Security Protocols	1	25%
II	Network Security & Access Control • Network Security Architecture, Firewalls and Packet Filtering, Intrusion Detection/Prevention Systems, Virtual Private Networks (VPNs), Wireless Security, Access Control Models	1	25%
III	System & Application Security • Operating System Security, Malware Analysis, Web Application Security, Database Security, Mobile Security, Cloud Security	1	25%
IV	Security Operations & Advanced Topics • Security Operations Center (SOC), Incident Response, Vulnerability Management, Ethical Hacking, Legal and Compliance, Emerging Technologies Security, Physical Security	1	25%

Textbooks:

- "Network Security Essentials: Applications and Standards" *William Stallings* (6th Edition) *Comprehensive coverage of network security protocols*
- "Computer Security: Principles and Practice" *William Stallings & Lawrie Brown* (4th Edition) *Excellent all-around cybersecurity textbook*

Reference Books:

- "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" *Dafydd Stuttard & Marcus Pinto* (2nd Edition) *Essential for web application security*
- "Hacking: The Art of Exploitation" *Jon Erickson* (2nd Edition) *Practical exploitation techniques*
- "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software" *Michael Sikorski & Andrew Honig Comprehensive malware analysis guide*

Online Resources:

- OWASP (Open Web Application Security Project)
- SANS Institute Reading Room
- NIST Cybersecurity Framework

- MITRE ATT&CK Framework
- Cybersecurity & Infrastructure Security Agency (CISA)
- Kali Linux Documentation
- NPTEL: "Information Security" by Prof. V. Kamakoti

Practical List:

Session 1: Wireshark packet analysis and protocol investigation
Session 2: Cryptographic implementation (AES, RSA, hashing)
Session 3: SSL/TLS analysis and certificate management
Session 4: Firewall configuration (iptables/Windows Firewall)
Session 5: Network scanning with Nmap and vulnerability scanning with OpenVAS
Session 6: Web application security testing with Burp Suite
Session 7: Password cracking techniques and defence
Session 8: Wireless security assessment
Session 9: Snort IDS rule creation and monitoring
Session 10: SIEM implementation with Splunk/ELK
Session 11: Incident response simulation
Session 12: Digital forensics with Autopsy/FTK

COURSE CODE: BTCSE705
COURSE NAME: ENTERPRENURSHIP AND PRODUCT MANAGEMENT
Course Objective

- Understand entrepreneurial mindset, innovation processes, and startup ecosystems.
- Apply lean startup methodology for product discovery and validation.
- Design and execute product development lifecycle from ideation to launch.
- Analyze business models, financial planning, and funding strategies for tech ventures.
- Develop go-to-market strategies, pricing models, and growth frameworks.
- Evaluate product metrics, user experience, and product-market fit.

Course Outcomes:

CO1	Analyze entrepreneurial opportunities and validate problem-solutionfit.
CO2	Apply lean startup methodology and design thinking for product discovery
CO3	Design and manage product development lifecycle using agile methodologies
CO4	Develop business models, financial projections, and funding strategies

Unit	Content	Credit	Weightage
I	Entrepreneurial Mindset & Opportunity Discovery	1	25%

	• Introduction to Entrepreneurship, Innovation and Creativity, Opportunity Identification, Idea Validation Framework, Tech Startup Ecosystems		
II	Product Discovery & Business Modelling • Customer Discovery, Market Analysis, Value Proposition Design, Business Model Development, Product-Market Fit	1	25%
III	Product Development & Agile Management • Product Strategy, Agile Product Development, User-Centered Design, Technical Product Management, Minimum Viable Product (MVP) Development	1	25%
IV	Go-to-Market & Growth Strategies • Financial Planning for Startups, Funding Strategies, Go-to-Market Strategy, Growth Frameworks, Legal and Operational Aspects, Scaling and Exit Strategies	1	25%

Textbooks:

- "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses" *Eric Ries Foundation for modern startup methodology*
- "Inspired: How to Create Tech Products Customers Love" *Marty Cagan* (2nd Edition) *Essential for product management in tech*

Reference Books:

- "Zero to One: Notes on Startups, or How to Build the Future" *Peter Thiel with Blake Masters Philosophical approach to building monopolies*
- "The Startup Owner's Manual: The Step-by-Step Guide for Building a Great Company" *Steve Blank & Bob Dorf Practical guide for customer development*
- "Hooked: How to Build Habit-Forming Products" *Nir Eyal Understanding user psychology and engagement*

Online Resources:

- Y Combinator Startup School (Free courses)
- Stanford's Entrepreneurship Corner (Video lectures)
- Harvard Business Review (Case studies)
- Product School Blog and courses
- AngelList Resources for startup founders
- NPTEL: "Entrepreneurship" by IIT Kharagpur
- Ministry of MSME, Government of India resources

SEMESTER -VIII

COURSE CODE: BTCSE801
COURSE NAME: ADVANCED ALGORITHMS
Course Objective

- Analyze advanced algorithmic paradigms beyond basic design techniques.
- Design approximation algorithms for NP-hard optimization problems.
- Implement randomized algorithms and analyze their probabilistic guarantees.
- Develop online algorithms and streaming algorithms for large-scale data.
- Apply advanced graph algorithms and computational geometry techniques.
- Evaluate algorithmic complexity using amortized, competitive, and parameterized analysis.

Course Outcomes:

CO1	Design and analyze approximation algorithms with proven performanceguarantees.
CO2	Implement randomized algorithms and analyze expected performance bounds
CO3	Develop online algorithms with competitive ratio analysis
CO4	Design algorithms for streaming and massive data processing

Unit	Content	Credit	Weightage
I	Approximation Algorithms • Introduction to Approximation Algorithms, Greedy Approximation Algorithms, LP-Based Approximation, Primal-Dual Algorithms, Advanced Approximation Techniques, Inapproximability	1	25%
II	Randomized Algorithms • Randomized Algorithm Basics, Randomized Data Structures, Randomized Graph Algorithms, Probabilistic Method, Markov Chains and Random Walks, Advanced Randomized Techniques	1	25%
III	Online & Streaming Algorithms • Online Algorithms Framework, Online Scheduling, Streaming Algorithms, Graph Streaming, Sliding Window Algorithms	1	25%
IV	Advanced Topics & Parameterized Algorithms • Parameterized Complexity, Amortized Analysis Advanced, Computational Geometry Algorithms, String Algorithms, Network Flow Advanced, Parallel Algorithms, External Memory Algorithms	1	25%

Textbooks:

- "Approximation Algorithms" *Vijay V. Vazirani Comprehensive coverage of approximation algorithms*
- "Randomized Algorithms" *Rajeev Motwani & Prabhakar Raghavan Classic reference for randomized algorithms*

Reference Books:

- "The Design of Approximation Algorithms" *David P. Williamson & David B. Shmoys Excellent for LP-based approximation techniques*
- "Parameterized Algorithms" *Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh Modern comprehensive parameterized algorithms book*
- "Data Streams: Algorithms and Applications" *S. Muthu Muthukrishnan Foundational streaming algorithms reference*

Online Resources:

- MIT 6.854: Advanced Algorithms (Course materials)
- NPTEL: "Advanced Algorithms" by Prof. Sundar Vishwanathan
- Tim Roughgarden's Algorithms Courses (Stanford/Coursera)
- Jeff Erickson's Algorithms Notes
- Competitive Programming Book by Antti Laaksonen
- Research Papers: Seminal papers in each area

COURSE CODE: BTCSE802
COURSE NAME: PARALLEL COMPUTING
Course Objective

- Understand parallel computing architectures, models, and performance metrics.
- Design and analyze parallel algorithms for shared and distributed memory systems.
- Implement parallel programs using MPI, OpenMP, and CUDA programming models.
- Optimize parallel applications for performance, scalability, and energy efficiency.
- Evaluate parallel algorithms using complexity measures and performance analysis tools.
- Design solutions for load balancing, synchronization, and communication in parallel systems

Course Outcomes:

CO1	Analyze parallel computing architectures and classify parallel systems using Flynn's taxonomy.
CO2	Design and analyze parallel algorithms using PRAM and work-span models
CO3	Implement shared memory parallel programs using OpenMP and P threads
CO4	Develop distributed memory parallel applications using MPI

Unit	Content	Credit	Weightage
I	Parallel Computing Foundations & Architectures • Introduction to Parallel Computing, Parallel Architectures, Performance Metrics and Laws, Parallel Algorithm Design Principles, Parallel Programming Models	1	25%
II	Shared Memory Parallel Programming • Thread-based Parallelism, OpenMP Programming Model, Task Parallelism in OpenMP, Advanced OpenMP Features, Performance Optimization for Shared Memory, Synchronization Primitives	1	25%
III	Distributed Memory Parallel Programming • Message Passing Interface (MPI), Advanced MPI Features, Parallel Algorithm Design for Distributed Memory, Performance Analysis and Optimization, Hybrid Programming	1	25%
IV	GPU Computing & Advanced Topics • GPU Architecture, CUDA Programming Model, CUDA Programming, CUDA Optimization Techniques, Advanced GPU Programming, Alternative GPU Programming Models, Parallel Patterns and Algorithms, Emerging Trends	1	25%

Textbooks:

- "Introduction to Parallel Computing" *Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar* (2nd Edition) *Comprehensive coverage of parallel computing concepts*
- "Programming Massively Parallel Processors: A Hands-on Approach" *David B. Kirk & Wen-mei W. Hwu* (4th Edition) *Excellent for GPU computing and CUDA programming*

Reference Books:

- "Parallel Programming in C with MPI and OpenMP" *Michael J. Quinn Practical approach to MPI and OpenMP programming*
- "Structured Parallel Programming: Patterns for Efficient Computation" *Michael McCool, James Reinders, Arch Robison Pattern-based approach to parallel programming*
- "CUDA by Example: An Introduction to General-Purpose GPU Programming" *Jason Sanders & Edward Kandrot Hands-on introduction to CUDA programming*

Online Resources:

- NVIDIA CUDA Documentation and Tutorials
- OpenMP Specifications and Examples
- MPI Forum Documentation
- LLNL Parallel Computing Tutorials

- NPTEL: "Parallel Computing" by Prof. Subodh Kumar
- MIT OpenCourseWare: Parallel Computing courses
- Intel Parallel Studio Documentation

Practical List :

- Session 1: OpenMP setup and basic parallel constructs
- Session 2: OpenMP work-sharing and synchronization
- Session 3: OpenMP tasks and advanced features
- Session 4: Pthreads programming basics
- Session 5: Performance analysis and optimization of shared memory programs
- Session 6: MPI setup and point-to-point communication
- Session 7: MPI collective operations
- Session 8: Parallel matrix multiplication using MPI
- Session 9: Hybrid MPI+Open MP programming
- Session 10: CUDA setup and basic kernel programming
- Session 11: CUDA memory optimization and shared memory
- Session 12: Advanced CUDA features (streams, events)

COURSE CODE: BTCSE803
COURSE NAME: ROBOTICS
Course Objective

- Understand robot kinematics, dynamics, and control fundamentals.
- Implement robot motion planning and navigation algorithms.
- Design perception systems using sensors and computer vision for robots.
- Develop robot software architectures and control systems.
- Apply machine learning techniques for robot perception and decision-making.
- Design and evaluate robotic systems for real-world applications.

Course Outcomes:

CO1	Analyze robot kinematics and dynamics using mathematical models.
CO2	Implement robot motion planning and navigation algorithms
CO3	Design perception systems using sensors and computer vision
CO4	Develop robot control systems and software architectures

Unit	Content	Credit	Weightage
I	Robot Fundamentals & Kinematics • Introduction to Robotics, Mathematical Foundations, Robot Kinematics, Mobile Robot Kinematics, Robot Dynamics (Introduction)	1	25%

II	Robot Perception & Sensors • Robot Sensors, Computer Vision for Robotics, Visual Odometry and SLAM, Depth Perception, Sensor Fusion	1	25%
III	Robot Planning & Control • Motion Planning, Mobile Robot Navigation, Robot Control, Reactive Behaviors, Multi-Robot Systems	1	25%
IV	Robot Intelligence & Applications • Robot Learning, Robot Operating System (ROS), Human-Robot Interaction, Robot Software Architecture, Emerging Applications, Ethics and Safety in Robotics	1	25%

Textbooks:

- "Introduction to Autonomous Robots" *Nikolaus Correll, Bradley Hayes, Christopher Heckman, & Alessandro Roncone Modern approach with practical implementation focus*
- "Probabilistic Robotics" *Sebastian Thrun, Wolfram Burgard, Dieter Fox Foundational text for perception and SLAM*

Reference Books:

- "Robotics, Vision and Control: Fundamental Algorithms in MATLAB" *Peter Corke* (2nd Edition) *Excellent for algorithms with practical code*
- "Modern Robotics: Mechanics, Planning, and Control" *Kevin M. Lynch & Frank C. Park Comprehensive mathematical treatment*
- "Introduction to AI Robotics" *Robin R. Murphy* (2nd Edition) *AI perspective on robotics*

Online Resources:

- Open Robotics (ROS Documentation)
- Stanford CS223A: Introduction to Robotics (Course materials)
- NPTEL: "Introduction to Robotics" by Prof. Asokan T
- Coursera: "Robotics Specialization" (University of Pennsylvania)
- Robotics Back-End (Tutorials and articles)
- Gazebo Simulator Tutorials
- Arduino/Raspberry Pi Robotics Projects

Practical List :

- Session 1: Robot simulation setup (Gazebo/ROS)
- Session 2: Forward kinematics implementation for serial manipulator
- Session 3: Inverse kinematics using numerical methods
- Session 4: Mobile robot kinematics simulation
- Session 5: Camera calibration and image processing with OpenCV
- Session 6: Feature detection and matching for visual odometry
- Session 7: LiDAR data processing and obstacle detection
- Session 8: EKF implementation for sensor fusion
- Session 9: A* path planning implementation
- Session 10: RRT path planning for manipulator