ST. FRANCIS DE SALES COLLEGE

A FRANSALIAN INSTITUTE OF HIGHER EDUCATION AUTONOMOUS

NAAC A GRADE ● AFFILIATED TO BANGALORE UNIVERSITY ● AICTE APPROVED ● 2(F) & 12 (B) RECOGNITION OF UGC ● ISO 9001:2015 CERTIFIED

Electronics City P.O. Bengaluru - 560 100, Karnataka, INDIA (+91) 8088140679 pro@sfscollege.in www.sfscollege.in

MCA [2 YEARS] Syllabus and Scheme

2024 Admission Onwards

BOARD OF STUDIES
[COMPUTER SCIENCE AND APPLICATIONS]

St. Francis de Sales College
[Autonomous]
Electronics City P.O. Bengaluru 560100
Karnataka, INDIA

TABLE OF CONTENTS

MEMBERS OF THE BOARD OF STUDIES

Sl. No.	Name	Designation
1.	Dr. S. Sivagami, Program In-charge and Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Chairperson
2.	Dr. Hanumanthappa M Senior Professor, Department of Computer Science, Bangalore University	University Nominee
3.	Dr. Sabeen Govind P V Assistant Professor, Rajagiri College of Social Sciences	External Expert
4.	Dr. Kousalya Govardhanan Professor & Dean of Research, Dayananda Sagar University Electronic City (Subject Expert nominated by the Academic Council)	External Expert
5.	Maria Joseph Frederic, Senior Manager, IBM ISL R&D	Industry Expert
6.	Mr. Phani Pramod, Senior Development Manager, Essbase and Database tools, Oracle	Industry Expert
7.	Ms. Umme Hermain Shaikh Associate Consultant, Tarento Technologies, Bengaluru	Alumni
8.	Ms. Sailaja M Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
9.	Ms. S. Annie Christella Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member

10.	Ms. Saranya C Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
11.	Ms. Thejaswi Nandyala Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
12.	Ms. Amruta Gadad Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
13.	Ms. Sathiya Priya Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
14.	Ms. Gowthami Gunasekar Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
15.	Ms. Arundhati Ghosh, Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
16.	Mr. Joseph Rajakumar, Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
17.	Mr. Kirubakaran, Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member
18.	Ms. Samadrita Chakraborty, Assistant Professor, St. Francis de Sales College (Autonomous), Electronic City, Bengaluru.	Member

ST. FRANCIS DE SALES COLLEGE (AUTONOMOUS)

ABOUT THE COLLEGE

St. Francis de Sales College (Autonomous), popularly known as SFS College, is one of the leading Institutions of Higher Education in Bengaluru, Karnataka. Founded in 2004 with the vision of Excellence, Efficiency, and Transformation, and the Mission of Love of God and Service to Humanity, the College is run by the Missionaries of St. Francis de Sales (MSFS) of the South West India Province, also known as Fransalians. The College is accredited with "A" grade by NAAC, approved by AICTE, recognized under 2(f) & 12(b) by UGC, and certified under ISO 9001:2015. Permanently affiliated to Bangalore University, the College offers several degree programs at the Bachelors, Masters, and Doctoral levels under various disciplines. In 2024, St. Francis de Sales College received the Autonomous status, and it remains as a center for quality education, equipping the students with the skills, knowledge, and values needed to excel and make a meaningful impact in the world.

VISION AND MISSION

VISION

Excellence, Efficiency and Transformation.

MISSION

Love of God and Service to Humanity.

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

The Computer Science and Applications Department is dedicated to advancing the understanding of computational systems and technologies through rigorous education, innovative research, and community engagement. The department offers a comprehensive curriculum that blends theoretical foundations with practical skills to prepare students for the rapidly evolving technology landscape. With a focus on problem-solving, software development, and cutting-edge research, the department strives to equip students with the tools and knowledge required to excel in a variety of computing fields.

VISION AND MISSION

VISION

Empowering through technology, innovation and expertise

MISSION

Leveraging computation knowledge to drive societal progress and student success.

ELIGIBILITY CRITERIA

A candidate with any degree of a minimum of 3 years duration (10+2+3) of Bangalore university or of any other University equivalent there in to with a minimum of 50% of marks in the aggregate of all subjects including languages, if any, provided further, that the candidate has studied Mathematics / Computer science /Business Mathematics / Statistics / Computer Applications / Electronics as a subject at PUC level or equivalent HSC (XII Standard) or at Degree level is eligible for admission to MCA Course. Relaxation to SC/ST, Group I be extended as per university norms.

PROGRAMME STRUCTURE AND DURATION

The programme is for Two (02) years consisting of Four Semesters altogether. A candidate shall complete his/her degree within Two (2) academic years from the date of his/her admission to the first semester. A Student who successfully completes Two (02) years of the programme will be awarded Master's in Computer Applications (MCA) by Bangalore University. The maximum period for completion of course shall be four years from the date of admission. To be eligible for the award of the MCA degree, a candidate shall have completed the scheme of training and passed in all subjects prescribed for the Course.

PROMOTION

A candidate who has obtained a minimum of 40% marks in all the Semesters in each subject shall be eligible for a pass and 50% of the aggregate inclusive of internal assessment marks obtained in all subjects put together. A candidate is allowed to carry over all previous uncleared (Failed) theory and practical papers to subsequent semesters from first to fourth semester.

PROGRAM OUTCOME (PO)

PO1	Computational Knowledge	Apply knowledge of computing fundamentals, computing specialization, mathematics, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements.
PO2	Problem Analysis	Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines.
PO3	Design /Development of Solutions	Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.
PO4	Conduct Investigations of Complex Computing Problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
PO5	Modern Tool Usage	Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of the limitations.
PO6	Professional Ethics	Understand and commit to professional ethics and cyber regulations, responsibilities, and norms of professional computing practice.
PO7	Life-long Learning	Recognize the need, and have the ability, to engage in independent learning for continual development as a computing professional.
PO8	Project management and finance	Demonstrate knowledge and understanding of the computing and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO9	Communication Efficacy	Communicate effectively with the computing community, and with society at large, about complex computing activities by being able to comprehend and write effective reports, design documentation, make effective presentations, and give and understand clear instructions.
PO10	Societal and Environmental Concern	Understand and assess societal, environmental, health, safety, legal, and cultural issues within local and global contexts, and the consequential responsibilities relevant to professional computing practice.
PO11	Individual and Team Work	Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary environments.
PO12	Innovation and Entrepreneurship	Identify a timely opportunity and using innovation to pursue that opportunity to create value and wealth for the betterment of the individual and society at large.

ACADEMIC YEAR 2024-25
PG CONTINUOUS INTERNAL ASSESSMENT

THEORY:

1. Continuous Internal Assessment (C1 & C2) – 30 marks
2. End Semester Examination – 70 marks

PRACTICAL:

1. Continuous Internal Assessment (C1 & C2) – 30 marks
2. End Semester Practical Examination – 70 marks

PROJECT / DISSERTATION: As per AICTE norms

Sl. No	Assessments	Components	Marks & Attendance	IA Marks
1.	**Attendance and Regularity**	**C1**	**10**	**10**
2.	**Quality of Work and Documentation**	**C1**	**10**	**10**
3.	**Presentation**	**C1**	**10**	**10**
			TOTAL	**30**

S.NO	ASSESSMENTS	COMPONENTS	MARKS & ATTENDANCE	IA MARKS
	THEORY AND PRACTICAL SUBJECTS			
1	**Unit Test (25% of Syllabus)**	**C1**	**25**	**2.5**
2	**Case Study / Assignment**	**C1**	**10**	**5**
3	**Seminar**	**C1**	**10**	**5**
3	**Mid Semester Examination (70% of Syllabus)**	**C2**	**70**	**10**
4	**Unit test II (25% of Syllabus covered after the MSE)**	**C1**	**25**	**2.5**
4	**Attendance** • 75.00-79.99% - 1 Mark • 80.00-84.99% - 2 Marks • 85.00-89.99% - 3 Marks • 90.00-94.99% - 4 Marks • 95.00-100.00% - 5 Marks	**C2**	**Minimum of 75%**	**5**
	Total			**30 marks**

GRADING SYSTEM

Table of Conversion of % Marks to grade point:

% Marks	Grade Point
96-100	10
91-95	9.5
86-90	9.0
81-85	8.5
76-80	8.0
71-75	7.5
66-70	7.0
61-65	6.5
56-60	6.0
51-55	5.5
46-50	5.0
41-45	4.5
40	4

Final Result/Grade Description:

Semester/ Programme % of Marks	Semester GPA/ Programme/ CGPA	Grade Alpha Sign	Result/Class Description
90.1-100	9.01-10.00	O	Outstanding
80.1-90.0	8.01-9.00	A+	First Class Exemplary
70.1-80.0	7.01-8.00	A	First Class Distinction
60.1-70.0	6.01-7.00	B+	First Class
55.1-60.0	5.51-6.00	B	High Second Class
50.1-55.0	5.01-5.50	C	Second Class
40.0-50.0	4.00-5.00	P	Pass Class
Below 40	Below 4.0	F	Re-Appear

EXTERNAL EVALUATION

THEORY COURSE

There shall be a written semester examination at the end of each semester for all theory courses of duration of 3 hours with maximum 70 marks. A question paper may contain short answer type and long essay type questions. The question paper pattern is as follows.

SECTIONS	TYPE OF QUESTIONS	MARKS	NUMBER OF QUESTIONS TO BE ANSWERED
A	CONCEPTUAL	6	5 OUT OF 8
B	ANALYTICAL AND PROBLEM SOLVING	10	4 OUT OF 6
TOTAL 70 MARKS			

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
MCA COURSE MATRIX AS PER 2024

SEMESTER I

		Total Teaching hours	Duration of Exam (hrs.)	Marks			Credits
Subjects	**Paper/Subject Code**			**IA**	**Uni. Exam**	**Total**	
Mathematical Foundations for Computer Science	24MCA11	60	3	30	70	100	4
Data Structures	24MCA12	60	3	30	70	100	4
Software Engineering	24MCA13	60	3	30	70	100	4
Database Management Systems	24MCA14	60	3	30	70	100	4
Computer Organization & Architecture	24MCA15	60	3	30	70	100	4
Data Structures Lab	24MCA16P	60	3	30	70	100	2
Database Management Systems Lab	24MCA17P	60	3	30	70	100	2
Computer Organization and Architecture Lab	24MCA18P	60	3	30	70	100	2
MOOC Course	24MCA19M	Minimum 4 Week	3	30	70	100	2
Total Credits							28

SEMESTER II

		Total Teaching hours	Duration of Exam (hrs.)	Marks			Credits
Subjects	**Paper/Subject Code**			**IA**	**Uni. Exam**	**Total**	
Object Oriented programming with Java	24MCA21	60	3	30	70	100	4
Computer Networks	24MCA22	60	3	30	70	100	4
Operating Systems	24MCA23	60	3	30	70	100	4
Design and Analysis of Algorithms	24MCA24	60	3	30	70	100	4
Artificial Intelligence	24MCA25	60	3	30	70	100	4
Employability and Skill Development	24MCA26	60	3	30	70	100	2
Java Programming Lab	24MCA27P	60	3	30	70	100	2
Artificial Intelligence Lab Using Python	24MCA28P	60	3	30	70	100	2
Internship	24MCA29I	60	3	30	70	100	2
Total Credits							28

24MCA11: MATHEMATICAL FOUNDATIONS FOR COMPUTER SCIENCE

Course Code	24MCA11	**Course Title**	Mathematical Foundation for Computer Science
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**INTRODUCTION 15 HOURS** Sets: Basic Concepts Relations: Binary relations, Equivalence relations and partition. Functions: Different types of functions, Composition and Inverse, Recursive and hashing functions. Mathematical Induction. Partial Ordering Relations Partially ordered set: Representation of Poset - Hasse Diagram, LUB, GLB, well ordered set, meet and join of elements. Lattices as partially ordered sets: Definition and basic properties, Lattices as algebraic systems, sub lattices. Basic Concepts of Automata Theory: Alphabets, Strings, Languages, DFA, NFA and their representations.		
II	**PROBABILITY 16 HOURS** Probability: The Concept of Probability-Sample Spaces, Probability as Relative Frequency, Axiomatic Definition of Probability, Properties of Probability, Additive Property, Conditional Probability, Multiplicative Law of Probability, Law of Total Probability, Bayes' Formula, Independent Events. Random Variables, Distribution Functions, Discrete Random Variables, Continuous Random Variables, Probability Mass Function and Probability Density Function, Expectation and Variance, Functions of Random Variables, Some important Probability Distributions: Discrete - Bernoulli Trials and Binomial distribution, Geometric distribution and Poisson distribution, Continuous - Uniform distribution, Normal distribution and Exponential distribution.		
III	**LOGIC 14 HOURS** Logic Mathematical logic, Logical operators – Conjunction, Disjunction, Negation, Conditional and biconditional. Truth tables. Equivalence formula, Tautology, methods of proof-direct, indirect, contradiction, equivalence and induction. Inference Theory, Validity by truth table, Rules of Inference. . Predicate calculus: Predicates , statement functions, variables and quantifiers, predicate formulas, free and bound variables, the universe of discourse.		

IV	**GRAPH THEORY 15 HOURS** Graph Theory Basic terminology: Different types of graphs – Directed and undirected, Simple, Pseudo, Complete, Regular, Bipartite. Incidence and degree, Pendant and Isolated vertex and Null graph. Isomorphism, Sub graphs, Walk, Path and Circuit, Connected and disconnected graphs and components, operations on graphs. Euler Graphs, Fleury's Algorithm, Hamiltonian circuits and paths. Traveling salesman problem. Matrix representation of graphs – Incidence and Adjacency matrices.

REFERENCE BOOKS:

1. Kenneth H. Rosen: **Discrete Mathematics and Its Applications**, 8th Edition, McGraw-Hill, 2019.
2. J.P. Tremblay, R. Manohar: **Discrete Mathematical Structures with Applications to Computer Science**, 1st Edition, McGraw-Hill, 2001.
3. Sheldon M. Ross: **A First Course in Probability**, 10th Edition, Pearson, 2019.
4. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: **Introduction to Automata Theory, Languages, and Computation**, 3rd Edition, Pearson, 2013.
5. Douglas B. West: **Introduction to Graph Theory**, 2nd Edition, Pearson, 2001.

COURSE OBJECTIVES:

Applying mathematical concepts: Students will learn to relate practical examples to the appropriate mathematical model and interpret the associated operations and terminology. Analyzing and solving problems: Students will learn to analyze and solve practical computing problems.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	**Understand and apply set theory, relations, and functions** to solve mathematical problems, including recursive and hashing functions, and represent partially ordered sets using Hasse diagrams.
CO2	**Analyze and solve problems related to probability theory**, including conditional probability, Bayes' formula, and probability distributions such as binomial, geometric, Poisson, normal, and exponential distributions.
CO3	**Apply principles of mathematical logic** including truth tables, logical operators, and rules of inference, to establish the validity of arguments and perform logical proofs using various methods.
CO4	**Explore the fundamental concepts of automata theory**, including deterministic and non-deterministic finite automata (DFA and NFA), and their application in formal language theory.
CO5	**Understand and apply graph theory concepts**, including graph types, Euler and Hamiltonian paths, matrix representations, and algorithms like Fleury's Algorithm, to solve real-world problems such as the traveling salesman problem.

TEACHING PEDOGOGY

Active Learning through Practical Application, Conceptual Understanding, Problem-Based Learning, Collaborative Learning, Use of Technology, Scaffolding and Differentiation, Assessment for Learning. Formative Assessment: Regular quizzes, problem-solving sessions, and practical tasks on key concepts such as Accrual Basis and Going Concern. This helps in tracking student progress and understanding. Summative Assessment: Evaluate students' ability to analyzing the problem and apply the algorithm.

SKILL DEVELOPMENT

1. Logical Reasoning and Analytical Skills
2. Problem-Solving Skills
3. Data Analysis and Modeling.
4. Algorithmic thinking.

24MCA12: DATA STRUCTURES

<table>
<tr><td><b>Course Code:</b></td><td>24MCA12</td><td><b>Course Title</b></td><td>Data Structures</td></tr>
<tr><td><b>Course Type</b></td><td>DSC</td><td><b>Contact Hours</b></td><td>4 Hours per Week <b>Total</b>: 60 Hours</td></tr>
<tr><td><b>Credit</b></td><td>4</td><td><b>Domain</b></td><td>COMPUTER SCIENCE</td></tr>
<tr><td colspan="4"><b>Syllabus</b></td></tr>
<tr><td>I</td><td colspan="3"><b>Fundamentals of Data Structures and Arrays 15 Hours</b>
Introduction to Data Structures: Data Types, Structures; Arrays: Polynomial Representations, Operations: Polynomial Addition, Multiplication, Sparse Matrices; Stack: Definition and Concepts, Stack Operations, Applications: Infix to Postfix Conversion, Evaluation of Arithmetic Expressions.</td></tr>
<tr><td>II</td><td colspan="3"><b>Queues, Dynamic Memory, and Linked Lists: 16 Hours</b>
Queues, Queue Representation, Circular Queue, Double-Ended Queue; Priority Queue: Implementation using Heap Sort; Dynamic Memory Allocation Functions: malloc, calloc, realloc, free; Linked Lists: Operations: Insertion, Searching, Removing, Updating, Sorting, Reversing; Polynomials: Representation, Addition, Multiplication using Linked Lists</td></tr>
<tr><td>III</td><td colspan="3"><b>Linear and Non-Linear Data Structures 13 Hours</b>
Linear Data Structures: Linked Stacks, Linked Queues, Circular Linked List, Double-Ended Queue, Doubly Linked List, Circular Doubly Linked List; Non-Linear Data Structures: Graphs: Representation (Adjacency Matrix, Adjacency List), Merits and Demerits; Searching: Linear Search, Binary Search</td></tr>
<tr><td>IV</td><td colspan="3"><b>Trees and Advanced Data Structures 16 Hours</b>
Trees: Basic Terminology, Binary Trees, Binary Search Trees, Binary Search Tree Operations: Insertion, Deletion, Searching, Traversal (In-Order, Pre-Order, Post-Order), Threaded Binary Tree: Operations, Balanced Trees: AVL Trees (Properties, Insertion, Deletion, Rotations) Advanced Data Structures: Red-Black Trees: Properties, B-Trees: Operations (Searching, Node Creation, Splitting, Insertion, Deletion), B+ Trees: Definition and Structure, Disjoint Sets: Operations, Linked List Representation, Disjoint-Set Forests<b>.</b></td></tr>
</table>

REFERENCE BOOKS:

1. Mark Allen Weiss, "Data Structures and Algorithm Analysis in C", 4th Edition, Pearson, 2022.
2. Reema Thareja, "Data Structures Using C", 3rd Edition, Oxford University Press, 2020.
3. Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms in Python", 2nd Edition, Wiley, 2020.
4. D.S. Malik, "Data Structures and Algorithms: A Modern Approach", 1st Edition, Cengage Learning, 2021.
5. Harry H. Chaudhary, "Data Structures and Algorithms with C++: Modern Approach for Beginners", 1st Edition, Infinite Study, 2021.

COURSE OBJECTIVES:

The course aims to introduce fundamental concepts of algorithms and data structures, focusing on their design, analysis, and implementation. It seeks to equip students with the skills to develop efficient algorithms and apply them to solve computational problems using the C programming language. Additionally, the course covers essential data structures (arrays, linked lists, stacks, queues, and trees) and fundamental algorithms for sorting, searching, and pattern matching, preparing students for real-world problem-solving and software development tasks.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	**Understand the fundamental concepts of algorithms**, including their design, growth of functions, and efficiency, as well as their importance in solving computational problems.
CO2	**Develop proficiency in basic programming concepts in C**, including syntax, control structures, data types, input/output operations, and the use of loops for iteration.
CO3	**Gain knowledge of fundamental data structures**, such as arrays, strings, and linked lists, and apply operations like traversal, insertion, and deletion to solve practical programming problems.
CO4	**Master the implementation and application of advanced data structures**, including stacks and queues, and their role in function call management, expression evaluation, and data organization.
CO5	**Analyze and implement basic sorting, searching, and pattern matching algorithms**, understanding their efficiencies and applying them to real-world problems through a mini-project.

TEACHING PEDAGOGY

The teaching pedagogy for **Algorithms and Basic Programming Concepts** combines lectures with hands-on programming sessions to introduce core concepts and reinforce them through practical implementation in C. Problem-solving exercises and collaborative learning encourage students to design, analyze, and implement algorithms, while visual demonstrations help clarify complex ideas like sorting and tree traversal. Finally, mini-projects and continuous assessments promote the application of learned concepts to real-world problems, fostering deeper understanding and skill development.

SKILL DEVELOPMENT

- Algorithmic Thinking
- Programming Proficiency in C
- Data Structure Manipulation
- Practical Problem-Solving

24MCA13: SOFTWARE ENGINEERING

Course Code:	24MCA13	**Course Title**	Software Engineering
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**Introduction to Software Engineering and Agile Methodologies (15 hours)** **Introduction to Software Engineering**: Definition, characteristics, software development life cycle (SDLC), software myths; **Software Process Models**: Waterfall model, Incremental model, Prototyping model, Spiral model, Rapid Application Development (RAD) model; **Agile Methodology**: Overview of Agile, Scrum framework (Scrum Master, Product Owner, Sprint cycles), Extreme Programming (XP), Lean Software Development; **Agile Metrics and Tools**: Velocity, burn-down charts, cumulative flow diagrams. Tools such as **JIRA**, **Trello**, and **Asana** for agile project tracking and management; **Requirements Engineering**: Functional and non-functional requirements, requirement elicitation techniques, requirement analysis and specification, use cases; **Project Planning and Risk Management**: Project management processes, estimation techniques (COCOMO, Function Point Analysis), risk management strategies.		
II	**Software Design, Architecture, and Agile Design Principles (15 hours)** **Design Concepts**: Design principles – Abstraction, refinement, modularity, cohesion, and coupling, functional independence; **Architectural Design**: Software architecture, architectural styles, component-based design, microservices architecture, design patterns; **Agile Design Principles**: Agile design practices like **refactoring**, simplicity, and emergent design; **Detailed Design**: Transaction and transformation mapping, refactoring of designs, use of design principles in Agile frameworks; **User Interface (UI) Design**: Basics of UI design, interface analysis, interface design steps, prototyping; **Unified Modeling Language (UML) Diagrams**: Use case diagrams, class diagrams, sequence diagrams, activity diagrams.		

III	**Software Quality, Agile Testing Strategies and Software Maintenance (15 hours)** **Software Quality Assurance (SQA)**: Quality concepts, SQA activities, ISO standards, CMMI, TQM, Six-Sigma; **Agile Testing Strategies**: Test-driven development (TDD), behavior-driven development (BDD), continuous testing in Agile environments; **Verification and Validation**: Reviews, inspections, walkthroughs, and audits; **Software Metrics and Measurements**: Process, project, and quality metrics. **Software Maintenance**: Types of maintenance (corrective, adaptive, perfective, preventive), maintenance process;
IV	**Re-engineering, Project Management, and Agile Tools (15 hours)** **Re-engineering & Reverse Engineering**: Concepts and processes; **Software Configuration Management**: Version control, change management, configuration audits, software versioning; **Agile Project Management Tools**: Introduction to Agile project management tools such as **JIRA**, **Git**, **Trello**, and other collaboration tools used in Agile environments; **Software Project Management**: Project planning, scheduling, estimation techniques (COCOMO, Function Point Analysis), risk management; **Quality Management**: Quality assurance, review techniques, product and process metrics, and quality models.

REFERENCE BOOKS:

1. "Software Engineering: A Practitioner's Approach" by Roger S. Pressman and Bruce R. Maxim 9th edition, 2020 (McGraw-Hill Education)(VitalSource)
2. "Software Engineering" by Ian Sommerville 10th edition, 2015
3. "Object-Oriented Software Engineering Using UML, Patterns, and Java" by Bernd Bruegge and Allen H. Dutoit 3rd edition, 2009
4. "Software Engineering Fundamentals" by Richard H. Thayer and Mark J. Christensen 1st edition, 2005
5. "Applied Software Project Management" by Andrew Stellman and Jennifer Greene 1st edition, **2005**

COURSE OBJECTIVES:

The course aims to provide students with a thorough understanding of software engineering principles and practices, particularly in agile development methodologies. It covers the fundamentals of software processes, requirements gathering, and the importance of teamwork in agile environments. Students will learn to apply UML and design models for software development and engage in code reviews to evaluate and enhance software quality. This course prepares students to navigate the complexities of modern software projects, emphasizing collaboration, time management, and quality assurance techniques.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Understand and Apply Agile Software Development Principles.
CO2	Analyze and Implement UML and Design Models.
CO3	Collaborate in Agile Teams to Address Teamwork and Role Schemes.
CO4	Evaluate Software Design through Code Reviews and Object-Oriented Design.
CO5	Apply effective time management and measurement techniques in software projects to enhance productivity and ensure quality outcomes.

TEACHING PEDAGOGY

Lecture-based instruction on Agile software development principles, process models, and teamwork dynamics in Agile environments. Lab sessions focused on practical application of UML modeling, design processes, and code reviews through group projects and collaborative exercises. Interactive workshops to develop skills in time management, measurement activities, and quality assurance techniques within software projects. Case studies and reflective practices to analyze real-world Agile software development scenarios and enhance understanding of team collaboration and leadership roles.

SKILL DEVELOPMENT

1. Proficiency in Agile software development methodologies and their application in various project environments.
2. Practical skills in UML design, object-oriented design concepts, and software architecture principles.
3. Hands-on experience in collaborative teamwork, role assignments, and effective communication in Agile teams.
4. Understanding and application of quality assurance practices, including test-driven development and code review processes.
5. Mastery of reflective practices and iterative development, enabling continuous improvement in software engineering projects.

24MCA14: DATABASE MANAGEMENT SYSTEMS

Course Code:	24MCA14	**Course Title**	Database Management Systems	
Course Type	DSC	**Contact Hours**	4 Hours per Week	**Total**: 60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE	
Syllabus				
I	**UNIT 1: DATABASES AND DATABASE USERS: 15 HOURS** Introduction to Databases, Characteristics of the Database Approach, Actors on the Scene and Workers Behind the Scene, Advantages of Using DBMS Approach, Brief History of Database Applications; Database System Concepts and Architecture: Data Models, Schemas, Instances, Three-Schema Architecture and Data Independence, Database Languages and Interfaces, The Database System Environment, Centralized and Client-Server Architectures, Classification of Database Management Systems.			
II	**UNIT 2: DATA MODELING USING ER MODEL: 14 HOURS** Using High-Level Conceptual Data Models for Database Design, Entity Types, Entity Sets, Attributes, and Keys, Relationship Types, Relationship Sets, Roles, and Structural Constraints, Weak Entity Types, Refining the ER Design, Company Database Diagrams, Naming Conventions and Design Issue, File Organization and Storage: Secondary Storage Devices, Heap Files, Sorted Files, Hashing Techniques, Single-Level Ordered Index, Multi-Level Indexes, Indexes on Multiple Keys, Other Types of Indexes.			
III	**UNIT 3: RELATIONAL ALGEBRA & SQL: 16 HOURS** Features, Codd's Rule, Structure of Relational Databases, Relational Algebra with Extended Operations, Modifications of Database, Relational Calculus (Idea), SQL: Basic Structure, Set Operations, Aggregate Functions, Null Values, Nested Subqueries, Derived Relations, Views, Join Relations, DDL in SQL, **MySQL**: Basic Commands, Data Types, CRUD Operations, MySQL: Introduction to Indexing, Joins, Views, and Subqueries, **MongoDB:** Basic Commands, Data Types, Collections, Documents, Queries, Indexes, Aggregation Framework.			
IV	**UNIT 4: TRANSACTION PROCESSING SYSTEM: 15 HOURS** Introduction to Transaction Processing, Transaction and System Concepts, Desirable Properties of Transactions (ACID), Transaction Support in SQL, Concurrency Control Techniques: Two-Phase Locking, Timestamp Ordering, Multi-Version, and Validation-Based Techniques, Recovery Techniques: Recovery Concepts, Recovery in Multi-Database Systems, Database Backup, and Recovery from Catastrophic Failures.			

REFERENCE BOOKS:

1. Elmasri and Navathe: Fundamentals of Database Systems, 7th Edition, Addison -Wesley, 2016.
2. Silberschatz, Korth and Sudharshan Data base System Concepts, 7th Edition, Tata McGraw Hill, 2019.
3. Alex Petrov: Database Internals: A Deep Dive into How Distributed Data Systems Work, 1st Edition, O'Reilly Media, 2019.
4. Martin Kleppmann: Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, 1st Edition, O'Reilly Media, 2017.
5. Ramez Elmasri and Shamkant Navathe: Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition, Pearson, 2016

COURSE OBJECTIVES

The course aims to introduce students to the foundational concepts of database systems, focusing on the design, implementation, and management of databases. It covers data modeling, relational database theory, SQL programming, and database integrity. Additionally, it explores advanced topics like transaction processing, concurrency control, and recovery techniques, preparing students to handle complex database systems in real-world applications.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	**Understand the core concepts of databases**, including database models, schemas, architectures, and the role of DBMS in data management.
CO2	**Design databases using the Entity-Relationship (ER) model**, defining entity sets, attributes, keys, relationships, and refining database designs.
CO3	**Apply relational algebra and SQL** for querying and managing relational databases, including creating, modifying, and managing database structures and data.
CO4	**Develop stored procedures, triggers, and cursors using PL/SQL**, and ensure database integrity through the use of rules and constraints.
CO5	**Implement transaction processing, concurrency control, and recovery techniques** to maintain the reliability and consistency of databases, especially in multi-database environments.

.

TEACHING PEDOGOGY

Lectures will introduce core concepts such as database models, SQL programming, and transaction processing, supplemented by real-world examples to illustrate their applications. Interactive lab sessions will allow students to engage in database design, implement SQL queries, and work with PL/SQL, reinforcing theoretical knowledge through practical exercises. Group projects will encourage collaboration in designing and managing a database system, while case studies will provide insights into database applications in various industries.

SKILL DEVELOPMENT

- **Database Design Skills**: Learn to design efficient and scalable databases using conceptual and logical data models, ensuring adherence to best practices in database design.
- **Proficiency in SQL and PL/SQL**: Gain hands-on experience in writing complex SQL queries, creating stored procedures, and managing database integrity using triggers and constraints.
- **Transaction Management and Concurrency Control**: Develop skills in handling transactions, ensuring consistency in multi-user environments, and applying concurrency control techniques.
- **Database Backup and Recovery**: Learn techniques for safeguarding databases, including implementing backup strategies and recovery methods for catastrophic failures.

24MCA15: COMPUTER ORGANIZATION & ARCHITECTURE

Course Code	24MCA15	**Course Title**	Computer Organization & Architecture
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**Unit 1:** **15 Hours** **I**nput-Output Organization & Memory Unit: Accessing Input/Output devices; Interrupts; Data transfer schemes - programmed I/O and DMA transfer; data transfer schemes for microprocessors. Memory Unit Memory Hierarchy; Primary memory, Secondary Memory: Magnetic Tape, Magnetic Disk, Optical disk, Magneto-Optical Disk; Concepts of auxiliary, Associative, Cache And Virtual Memory, DMA, DMA Transfer modes, sequential access, direct access storage devices		
II	**Unit 2:** **16 Hours** Comparative Study of 8086 and 8088 : Evolution from 8080/8085 to 8086, Evolution from 8086 to 8088, 8086 Microprocessor: Pin diagram of 8086, Signal group of 8086, Internal organization of 8086, 8088 Microprocessor and its basic architecture, Pentium Processor: History, Block diagram, Dual Core Processor.		
III	**Unit 3:** **14 Hours** Transfer And Micro-Operations: Register Transfer Language, Register Transfer, Bus and Memory Transfers, Arithmetic Micro-Operations, Logic Micro-Operations, Shift Micro-Operations, Arithmetic logic shift unit. Micro-programmed Control: Control Memory, Address Sequencing, Micro-Program example, Design of Control Unit. Input Output: I/O interface, Programmed IO, Memory Mapped IO, Interrupt Driven IO, DMA. Instruction level parallelism: Instruction level parallelism (ILP)-overcoming data hazards, limitations of ILP.		
IV	**Unit 4:** **15 Hours** Multi-Processor Organization & Pipelining: Parallel Processing, Concept and Block Diagram, Types (SISD, SIMD, Interconnect network, MIMD, MISD), Future Directions for Parallel Processors, Performance of Processors, Pipelining: Data Path, Time Space Diagram, Hazards. Instruction Pipelining, Arithmetic Pipelining		

COURSE OBJECTIVES:

This course aims to provide a comprehensive understanding of the fundamental concepts of computer organization and architecture, including number systems, digital logic circuits, micro-operations, memory systems, and processor architectures. By the end of the course, students will have practical skills in handling 8085 assembly language programming and will understand the relevance of advanced processor and memory architectures in modern computing.

REFERENCE BOOKS:

1. Hayes, J.P., Computer Architecture and Organization, McGraw Hill (1998), 3rded.
2. William Stallings, "Computer Organization and Architecture designing for performance", 10th edition, Pearson(2016)
3. Subrata Ghoshal, "Computer Architecture And Organization", Pearson India(2011).
4. Andrew S. Tanenbaum " Structured Computer Organization", 5th edition, Pearson Education Inc(2006).
5. Carl Hamacher, Zvonks Vranesic,SafeaZaky, "Computer Architecture And Organization", 5 th edition McGraw Hill New Delhi,India(2002).
6. Mano M Morris, "Computer System Architecture", 3rd edition Pearson India(2019)

COURSE OUTCOME

CO CODE	COURSE DESCRIPTION
CO1	Understand the knowledge in **I**nput-Output Organization & Memory Unit
CO2	Apply knowledge of 8085 ,8086 and 8088 architecture to write, assemble, and execute basic assembly language programs, demonstrating proficiency in instruction cycles and addressing modes.
CO3	Explain the basic and advanced processor architectures (CISC, RISC, SIMD, pipelining), and differentiate between them based on instruction sets and processing efficiencies.
CO4	Execute register transfer, arithmetic, logic, and shift micro-operations, and manage input-output interfacing using programmed and interrupt-driven methods.
CO5	Explain the basic and advanced processor architectures about Multi-Processor Organization & Pipelining

TEACHING PEDAGOGY

Lecture-based instruction to introduce theoretical concepts and architectures. Problem-solving and practical exercises focusing on digital arithmetic, micro-operations, and instruction-level parallelism. Group discussions and case studies on advanced topics like SIMD, pipelining, and parallel processing.

SKILL DEVELOPMENT

- Mastery of number system conversion and binary arithmetic operations.
- Practical skills in designing and simplifying digital logic circuits for computing purposes.
- Ability to differentiate and optimize processor design and memory architecture for enhanced computing performance.
- Hands-on experience in 8086 assembly language programming.
- Understanding of instruction-level parallelism and pipelining concepts for performance enhancement in multi-core and parallel computing systems.

24MCA16: DATA STRUCTURES LAB

Course Code:	24MCA16P	**Course Title**	Data Structures Lab
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	2	**Domain**	COMPUTER SCIENCE
Syllabus			

LIST OF DATA STRUCTURES LAB PROGRAMS

1. Program to represent Linear Search and Binary Search.
2. Program to represent sorting procedures (Selection Sort, Bubble Sort, and Insertion Sort).
3. Polynomial addition using arrays.
4. Sparse matrix manipulation using arrays.
5. Program to allocate two-dimensional arrays dynamically.
6. Program to demonstrate the use of realloc().
7. Stack using arrays.
8. Queue using arrays.
9. Circular Queue using arrays.
10. Program to represent Singly Linked List.
11. Program to represent Doubly Linked List.
12. Program to represent Circular Linked List.
13. Polynomial addition using linked lists.
14. Program to represent a Queue using linked lists.
15. Program for a Binary Search Tree using recursion.
16. Program for Binary Search Tree Traversals (without recursion).
17. Program to represent a Graph using arrays.
18. Program for Infix to Postfix conversion.
19. Program for Evaluation of Postfix Expressions.
20. Program to represent a Graph using linked lists.

COURSE OBJECTIVES:
The course aims to provide students with a comprehensive understanding of fundamental data structures and their practical applications. It covers array, linked list, stack, queue, tree, and graph operations, along with sorting and searching algorithms. Students will gain hands-on experience in implementing and manipulating data structures, as well as analyzing their performance. This course prepares students to design, optimize, and apply data structures to solve complex computational problems efficiently.

COURSE OUTCOME	
COURSE CODE	COURSE DESCRIPTION
CO1	Students will understand and implement recursive algorithms like factorial computation and Fibonacci sequence generation.
CO2	Students will gain the ability to perform basic operations on integers and strings, such as swapping values, counting digits, and manipulating strings without built-in functions.
CO3	Students will learn to perform fundamental operations like traversal, insertion, and deletion on arrays and linked lists (both singly and doubly linked).
CO4	Students will implement and work with data structures such as stacks, queues, and binary trees, including various tree traversal methods.
CO5	Students will implement sorting and searching algorithms and analyze their performance

TEACHING PEDAGOGY

Lecture-based instruction to introduce the concepts of arrays, linked lists, stacks, queues, trees, and sorting/searching algorithms. Problem-solving and practical exercises focusing on implementing and manipulating these data structures in C. Group discussions and case studies on real-world applications like text editors and inventory management systems using arrays and linked lists.

SKILL DEVELOPMENT

1. Mastery of basic algorithmic thinking and problem-solving in C.
2. Practical skills in implementing and manipulating arrays, linked lists, and queues.
3. Proficiency in stack operations using both arrays and linked lists.
4. Hands-on experience in implementing binary trees and binary search trees.
5. Understanding of sorting and searching algorithms with performance optimization.

24MCA17: DATABASE MANAGEMENT SYSTEMS LAB

Course Code:	24MCA17P	**Course Title**	Database Management Systems Lab
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	2	**Domain**	COMPUTER SCIENCE
Syllabus			

List of Database Management Systems Lab Programs

1. Write a MySQL script to create a schema with tables, applying constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, and NOT NULL.
2. Develop a MySQL script to create tables with data types like VARCHAR, INT, DATE, etc.
3. Write a MySQL script to create a new database schema and assign appropriate permissions to users (e.g., GRANT, REVOKE).
4. Write a MySQL program to create a table that reflects different data types and applies constraints such as CHECK and DEFAULT.
5. Write a MySQL script to demonstrate how to drop a table.
6. Write a MySQL script to modify an existing table by adding or modifying columns (e.g., ALTER TABLE).
7. Write a MySQL script to drop a specific column from an existing table.
8. Write a MySQL script to demonstrate basic SQL queries such as SELECT, DISTINCT, WHERE
9. Write a MySQL script to demonstrate set operations like UNION, INTERSECT, and EXCEPT
10. Write a MySQL script to demonstrate the use of nested queries (e.g., subqueries in SELECT, WHERE, and FROM).
11. Write a MySQL script that demonstrates the use of the EXISTS function to test the existence of rows in subqueries.
12. Write a MySQL program to handle NULL values, including filtering for NULL in queries.
13. Write a MySQL script to demonstrate the use of aggregate functions like COUNT, SUM, AVG, MIN, and MAX.
14. Write a MySQL script to demonstrate the use of GROUP BY and HAVING for grouping and filtering query results.
15. Write a MySQL script to sort query results using the ORDER BY clause and perform basic arithmetic operations within queries.
16. Write a MongoDB script to create a collection and insert documents with various fields, including nested fields.
17. Write a MongoDB script to demonstrate CREATE, READ, UPDATE, and DELETE operations on a collection.
18. Write a MongoDB script to create indexes on a collection and demonstrate queries that benefit from these indexes.
19. Write a MongoDB script to demonstrate the use of the aggregation pipeline, including $group, $match, and $sum.
20. Write a MongoDB script to use change streams to monitor changes in a collection (equivalent to triggers in relational databases).

COURSE OBJECTIVES:
The course aims to provide students with a comprehensive understanding of database management systems and their practical applications. It covers database design using E-R modeling, relational schema conversion, and SQL-based database creation, manipulation, and querying. Students will gain hands-on experience in writing optimized SQL queries, managing transactions, and performing database backup and restoration. This course prepares students to design, implement, and maintain efficient and secure databases for real-world applications.

COURSE OUTCOME

COURSE CODE	COURSE DESCRIPTION
CO1	Design and implement E-R models, and convert them into relation tables for real-world scenarios.
CO2	Create, modify, and manipulate databases, tables, and records using basic and advanced SQL commands.
CO3	Apply SQL to query databases using aggregate functions, GROUP BY, HAVING, and EXISTS clauses.
CO4	Perform advanced SQL operations such as view creation, transactions, and database backup/restore.
CO5	Analyze and optimize SQL queries to retrieve specific information based on complex conditions in large datasets.

TEACHING PEDAGOGY

Lecture-based instruction to introduce the concepts of database management systems, E-R modeling, and SQL. Practical exercises and lab sessions focused on SQL query formulation and optimization. Group discussions on real-world scenarios like bank and college systems, emphasizing normalization, relational integrity, and constraints.

SKILL DEVELOPMENT

1. Mastery of E-R modeling and converting relational models into tables.
2. Proficiency in using SQL for database creation, manipulation, and retrieval.
3. Ability to handle database backup, restoration, and transaction control.
4. Hands-on experience in writing optimized queries using complex SQL clauses.
5. Understanding of views, indexing, and relational constraints for efficient database management

24MCA18: COMPUTER ORGANIZATION AND ARCHITECTURE LAB

Course Code:	24MCA18P	**Course Title**	Computer Organization and Architecture Lab
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	2	**Domain**	COMPUTER SCIENCE

Syllabus

LIST OF COMPUTER ORGANISATION AND ARCHITECTURE LAB PROGRAMS

1. HDL introduction
2. Realization of a Boolean Function. Minimize using K map and realize the same using truth table
3. Realize NAND and NOR Gate as universal gate
4. Design Half Adder and Full Adder
5. Design a Full Adder/ Subtractor using 2 half adder/ subtractor
6. Design Half Subtractor and Full Subtractor
7. Design 4 bit parallel Adder Subtractor Composite unit using IC7483 and 7486
8. Design 8:1 Multiplexer using two 4:1 Multiplexer
9. Implement logic function using Multiplexer.
10. 8-bit Addition, Multiplication, Division
11. 8-bit Register design
12. Memory unit design and perform memory operations.
13. 8-bit simple ALU design
14. 8-bit simple CPU design
15. Interfacing of CPU and Memory

COURSE OBJECTIVES:

The course aims to provide students with a thorough understanding of digital logic design, arithmetic units, and basic computer organization. It covers the design and implementation of logic gates, adders, subtractors, multiplexers, and memory units using hardware description languages (HDL) and integrated circuits. Students will gain hands-on experience in designing 8-bit arithmetic units, ALUs, and CPUs, along with CPU-memory interfacing. This course prepares students to design, simulate, and analyze fundamental digital components and architectures used in modern computers.

COURSE OUTCOME	
COURSE CODE	COURSE DESCRIPTION
CO1	Implement basic digital logic functions and Boolean algebra using HDL and truth tables.
CO2	Design and realize combinational circuits such as adders, subtractors, and multiplexers.
CO3	Develop arithmetic units like 8-bit adders, multipliers, and registers using integrated circuits.
CO4	Design and implement memory units and simulate basic memory operations.
CO5	Build and interface a simple 8-bit CPU with memory, understanding the fundamental components of a processor.

TEACHING PEDAGOGY

Lecture-based instruction to introduce digital logic design, Boolean algebra, and computer architecture fundamentals. Lab sessions and practical exercises will focus on designing and simulating digital components using HDL and hardware. Group discussions and problem-solving on optimizing logic circuits and interfacing memory units with CPUs. Case studies on modern computer architectures to understand the practical applications of digital design.

SKILL DEVELOPMENT

1. Mastery of Boolean logic, K-map simplifications, and truth table realizations.
2. Practical skills in designing and simulating combinational and arithmetic circuits like adders, multiplexers, and ALUs.
3. Proficiency in using HDL to design and test digital components.
4. Hands-on experience in building memory units and interfacing them with CPUs.
5. Understanding of basic CPU design and memory interfacing, preparing students for advanced computer architecture.

24MCA21: OBJECT ORIENTED PROGRAMMING WITH JAVA

Course Code:	24MCA21	**Course Title**	Object Oriented Programming with Java
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**Basics of Java** **15 HOURS** Java - What, Where and Why?, History and Features of Java, Internals of Java Program, Difference between JDK,JRE and JVM, Internal Details of JVM, Variable and Data Type, Unicode System, Naming Convention. OOPS Concepts: Advantage of OOPs, Object and Class, Method Overloading, Constructor, static variable, method and block, this keyword, Inheritance (IS-A), Aggregation and Composition(HAS-A), Method Overriding, Covariant Return Type, super keyword, Instance Initializer block, final keyword, Runtime Polymorphism, static and Dynamic binding, Abstract class and Interface, Down casting with instance of operator ,Package and Access Modifiers, Encapsulation, Object class, Object Cloning, Java Array, Call By Value and Call By Reference.		
II	**CORE JAVA FEATURES** **15 HOURS** Core java Features: String Handling, Exception Handling, Nested classes, Packages and Interfaces. Multithreaded Programming – synchronization, Input/Output – Files – Director, Utility Classes, Generics, Generic Class, Generic methods. Serialization: Serialization & Deserialization, Serialization with IS-A and Has-A, Transient keyword. Networking: Socket Programming, URL class, Displaying data of a web page, Inet Address class, Datagram Socket and Datagram Packet, Two way communication		
III	**JDBC** **14 HOURS** JDBC: - Overview, JDBC implementation, Connection class, Statements, Catching Database Results, handling database Queries. Error Checking and the SQL Exception Class, The SQL Warning Class, JDBC Driver Types, ResultSetMetaData, using a Prepared Statement, Parameterized Statements, Stored Procedures, Transaction Management. Collection: Collection Framework, ArrayList class, LinkedList class, ListIterator interface, HashSet class.		
IV	**Overview of JavaFX**: **16 HOURS** Introduction to JavaFX and its architecture; Creating JavaFX Applications, Java FX main application class; UI Controls: Working with buttons, text fields, labels, and other UI controls Layouts: VBox, HBox, GridPane; Event Handling: user interactions and events; CSS Styling; Introduction to FXML: Understanding its role in simplifying JavaFX UI design. FXML Structure: syntax and organization of FXML files. Controller Integration: Connecting FXML with Java code.		

REFERENCE BOOKS:

1. Herbert Schildt, "Java: The Complete Reference," 12th Edition, McGraw Hill (2021).
2. Bruce Eckel, "Thinking in Java," 4th Edition, Prentice Hall/Pearson Education (2006).
3. Ken Arnold and James Gosling, "The Java Programming Language," 4th Edition, Addison-Wesley (2005).
4. Maydene Fisher, Jon Ellis, and Jonathan Bruce, "JDBC API Tutorial and Reference," 3rd Edition, Addison-Wesley (2001).
5. Joe Wigglesworth and Paula McMillan, "Java Programming: Advanced Topics," 5th Edition, Cengage Learning (2011).

COURSE OBJECTIVES:

The course aims to provide students with a comprehensive understanding of Java programming, covering the basics of the language, object-oriented programming (OOP) principles, and core Java features. It introduces students to advanced topics such as exception handling, multithreading, networking, and database connectivity using JDBC. Additionally, students will explore Java's GUI components using AWT and Swing, preparing them to design and implement interactive applications in Java.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Ability to solve problems using only pure object-oriented concepts
CO2	Make decision to solve a problem using package, library and threads Handling Errors and Exceptions
CO3	Able to develop networking applications
CO4	Ability to design and develop database applications
CO5	Ability to utilize graphical user interface (GUI) components like AWT and Swing to design interactive applications.

TEACHING PEDAGOGY

Lecture-based instruction to introduce the theoretical concepts of Java programming, OOP principles, and core Java features. Practical lab sessions and coding exercises to implement these concepts, with a focus on real-world application development. Group discussions on advanced topics like multithreading, JDBC, and GUI development using AWT and Swing. Case studies and hands-on projects involving database-driven applications and network programming.

SKILL DEVELOPMENT

1. Proficiency in object-oriented programming using Java.
2. Practical skills in handling exceptions, multithreading, and file I/O operations.
3. Hands-on experience with database connectivity and query execution using JDBC.
4. Ability to design user interfaces using AWT and Swing components.
5. Mastery of Java’s collection framework and its efficient use in software development.

24MCA22: COMPUTER NETWORKS

Course Code:	24MCA22	**Course Title**	Computer Networks
Course Type	DSC	**Contact Hours**	4 Hours Per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**INTRODUCTION 15 HOURS** Introduction: Data Communications, Computer Networks, Network Layering-OSI reference Model, TCP-IP Protocol Suite. Physical Layer:Data and Signals, Periodic Analog Signals, Digital Signals, Transmission Impairment, Data rate Limits. Digital-to-Digital Conversion, Analog-to-Digital Conversion, Digital-to-Analog Conversion, Analog-to-Digital Conversion.		
II	**PHYSICS LAYER 14 HOURS** Physical Layer: Transmission and Switching Transmission Modes, Transmission media- Guided, unguided media. Multiplexing, Switching-Circuit Switching, packet switching		
III	**DATA LINK LAYER 16 HOURS** Data Link Layer: Nodes and Links, Link-Layer Addressing, error Detection and Correction- Block coding, Cyclic Codes, Checksum, Forward Error Correction, Simple, Stop-and-wait, Go-back-N, Selective Repeat Media Access Control: Random Access-ALOHA, CSMA, CSMA/CD, CSMA/CD, Controlled Access, Channelization-FDMA, TDMA, CDMA		
IV	**NETWORK LAYER 15 HOURS** Network Layer: Services, Routing Algorithms: Distance Vector, Link State, Path Vector, and Unicast Routing Algorithms. Multicasting Basics: Addresses, Delivery at Data Link Layer, Multicast Forwarding, Two Approaches to Multicasting. IP Addressing, Classes, Sub netting		

REFERENCE BOOKS:

1. Behrouz A. Forouzan, "Data Communications and Networking," 5th Edition, McGraw Hill, 2013.
2. Andrew S. Tanenbaum, "Computer Networks," 5th Edition, Prentice-Hall.
3. William Stallings, "Data and Computer Communications," 8th Edition, Pearson.
4. James F. Kurose and Keith W. Ross, "Computer Networking: A Top-Down Approach," 6th Edition, Pearson, 2012.
5. Larry L. Peterson and Bruce S. Davie, "Computer Networks: A Systems Approach," 5th Edition, Morgan Kaufmann, 2011.

COURSE OBJECTIVES:

The course aims to provide students with a deep understanding of data communication principles and computer networking. It covers network layering concepts with a focus on the OSI reference model and the TCP/IP protocol suite. The course explores physical layer techniques for signal transmission and conversion, as well as error detection, correction methods, and routing algorithms at the network layer. Students will gain practical knowledge of how data is transmitted, controlled, and routed in modern networks.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Understand data communication principles, network models (OSI, TCP/IP), and their layer interactions.
CO2	Apply physical layer concepts and data link layer techniques to design and analyze network systems.
CO3	Analyze network protocols and routing algorithms to optimize network performance.
CO4	Design network solutions using physical layer technologies, data link protocols, and network layer routing, including IP addressing and multicasting.
CO5	Troubleshoot and secure networks using encryption, authentication, and firewall technologies for data integrity and security.

TEACHING PEDAGOGY

Lecture-based instruction on data communication principles, networking models, and the functioning of each network layer. Lab sessions focused on simulating data transmission techniques, multiplexing, and switching, using network simulation tools. Group discussions on error correction methods, media access control, and routing algorithms. Case studies to explore real-world scenarios of network layer functions and routing across different network architectures.

SKILL DEVELOPMENT

2. Proficiency in understanding network models, including OSI and TCP/IP.
3. Practical skills in signal conversion, transmission media, and switching techniques.
4. Hands-on experience with data link layer error correction and media access protocols like ALOHA, CSMA/CD.
5. Understanding and implementation of routing algorithms and IP addressing schemes.
6. Mastery of network layer services, including routing, multicast forwarding, and subnetting.

24MCA23: OPERATING SYSTEMS

Course Code:	24MCA23	**Course Title**	Operating Systems
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**INTRODUCTION TO OPERATING SYSTEM 15 HOURS** Introduction: Computer System Organization, Architecture, Structure, Operations, Process Management, Memory Management, Storage Management, Kernel Data Structures, Computing Environments. Operating System Structures: Services, System Calls, Types, Operating System Structure, System Boot. Processes: Process Concept, Scheduling, Operations, Interprocess Communication. Multithreaded Programming: Multicore Programming, Multithreading Models.		
II	**PROCESS SYNCHRONIZATION 14 HOURS** Process Synchronization: The Critical-Section Problem, Peterson's Solution, Synchronization Hardware, Mutex Locks, Semaphores, Classic Problems of Synchronization, Monitors, Synchronization Examples. Process Scheduling: Criteria, Scheduling Algorithms, Multi-Processor Scheduling, Real-time CPU Scheduling. Deadlocks: System model, Characterization, Methods for handling deadlocks, Deadlock Prevention, Avoidance, Detection and Recovery from deadlock.		
III	**MEMORY MANAGEMENT STRATEGIES 15 HOURS** Background, Swapping, Contiguous Memory Allocation, Segmentation, Paging, Structure of the Page Table. Virtual Memory Management: Demand Paging; Copy-on-Write, Page Replacement; Allocation of Frames; Thrashing, Memory-Mapped Files, Allocating Kernel Memory. File System: File Concept, Access Methods, Directory and Disk Structure, Protection. File-System Implementation: Structure, File-System and Directory Implementation, Allocation Methods, Free Space Management, Efficiency and Performance, Recovery. Mass-Storage Structure: Overview, Disk Scheduling, Disk Management.		
IV	**PROTECTION 16 HOURS** Protection: Goals, Principles, Domain of Protection, Access Matrix, Implementation of the Access Matrix, Access Control, Revocation of the Access Rights. Virtual Machines: Building Blocks, Types of VMs and their implementations. Distributed Systems: Advantages, Types of Network based OS, Robustness, Design Issues, Distributed File Systems. Database Operating Systems: Requirements of Database OS – Transaction process model – Synchronization primitives - Concurrency control algorithms. Mobile Operating Systems: ARM and Intel architectures - Power Management - Mobile OS Architectures - Underlying OS - Kernel structure and native level programming - Runtime issues-Approaches to power management Case Studies: The Linux System.		

REFERENCE BOOKS:

1. Abraham Silberschatz, Peter B. Galvin, Greg Gagne: **Operating System Concepts**, 10th Edition, Wiley, 2018.
2. William Stallings: **Operating Systems: Internals and Design Principles**, 9th Edition, Pearson, 2018.
3. Andrew S. Tanenbaum, Herbert Bos: **Modern Operating Systems**, 4th Edition, Pearson, 2015.
4. Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau: **Operating Systems: Three Easy Pieces**, 1st Edition, Arpaci-Dusseau Books, 2018.
5. Thomas Anderson, Michael Dahlin: **Operating Systems: Principles and Practice**, 2nd Edition, Recursive Books, 2014.

COURSE OBJECTIVES:

The course aims to provide students with a comprehensive understanding of the fundamental concepts, structures, and mechanisms of operating systems. It covers process and memory management, synchronization, scheduling, file systems, and security, while also addressing advanced topics like virtual machines, distributed systems, database operating systems, and mobile OS architectures. The course prepares students to design, analyze, and optimize operating systems in various computing environments.

COURSE OUTCOME

CO CODE	COURSE DESCRIPTION
CO1	**Understand the core concepts of operating systems**, including architecture, process management, memory management, and storage management, and their role in computer system organization.
CO2	**Analyze and implement synchronization mechanisms**, such as semaphores, mutex locks, and monitors, to solve the critical-section problem and classic synchronization challenges.
CO3	**Evaluate different process scheduling algorithms** and apply them to manage processes and CPU resources efficiently, while understanding techniques for deadlock prevention and recovery.
CO4	**Demonstrate knowledge of memory management strategies**, including paging, segmentation, virtual memory, and demand paging, and their impact on system performance and efficiency.
CO5	**Explore advanced topics in operating systems**, such as distributed systems, database operating systems, virtual machines, and mobile OS architectures, understanding their design, challenges, and applications.

TEACHING PEDOGOGY

The teaching pedagogy for this **Operating Systems** syllabus combines theoretical instruction with hands-on practical learning. Lectures will cover core concepts such as process management, memory management, synchronization, and virtual memory, emphasizing both foundational knowledge and advanced topics like distributed systems and mobile OS architectures.

SKILL DEVELOPMENT

- **System-Level Problem-Solving**: Develop the ability to design and implement solutions to process management, synchronization, and memory management problems in operating systems.
- **Advanced Resource Management**: Gain expertise in handling deadlocks, process scheduling, and virtual memory management techniques, improving system efficiency.
- **OS Security and Protection**: Learn to implement protection mechanisms and access control in operating systems, ensuring secure and robust computing environments.
- **Hands-on Experience with OS Case Studies**: Apply theoretical knowledge through practical case studies, especially Linux, to understand real-world operating system implementations and optimizations.

24MCA24: DESIGN AND ANALYSIS OF ALGORITHMS

Course Code:	24MCA24	**Course Title**	Design and Analysis of Algorithms
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**INTRODUCTION 12 Hours** What is an Algorithm? Fundamentals of Algorithmic Problem Solving. FUNDAMENTALS OF THE ALGORITHMS EFFICIENCY: Analysis Framework, Asymptotic Notations and Standard notations and common functions, Mathematical Analysis of Non-recursive and Recursive Algorithms.		
II	**BRUTE FORCE 16 HOURS** Background, Selection Sort, Brute-Force String Matching. TSP DIVIDE AND CONQUER: General method, Recurrences: The substitution method, The recursion-tree method, The master method, Merge sort, Quick sort, Binary Search, Multiplication of large integers, Case study: Strassen's Matrix Multiplication. BRUTE FORCE: Background, Selection Sort, Brute-Force String Matching. TSP DIVIDE AND CONQUER: General method, Recurrences: The substitution method, The recursion-tree method, The master method, Merge sort, Quick sort, Binary Search, Multiplication of large integers, Case study: Strassen's Matrix Multiplication		
III	**DECREASE & CONQUER 14 HOURS** DECREASE & CONQUER: General method, Insertion Sort, Graph algorithms: Depth First Search, Breadth First Search, Topological Sorting TRANSFORM AND CONQUER: Case study: Heaps and Heap sort. TIME AND SPACE TRADEOFFS: Input Enhancement in String Matching: Horspool's algorithm, Hashing: Open and Closed hashing.		
IV	**GREEDY TECHNIQUE 16 HOURS** GREEDY TECHNIQUE: General method of Greedy technique, Single-Source Shortest Paths: General method, The Bellman-Ford algorithm, Single-Source Shortest Paths in DAGs, Dijkstra's Algorithm. Minimum Spanning Trees: Prim's Algorithm, Optimal Tree problem: Huffman Trees; Case study: Kruskal's Algorithm. Fractional Problem. DYNAMIC PROGRAMMING: LIMITATIONS OF ALGORITHMIC POWER P, NP and NP-complete problems, BACKTRACKING: General method, N-Queens problem, Subset-sum problem. BRANCH AND BOUND: General method, Travelling Salesman problem, Approximation algorithms for TSP. Case study: Knapsack Problem.		

REFERENCE BOOKS:

1. **Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein**: *Introduction to Algorithms*, 4th Edition, MIT Press, 2022.
2. **Jon Kleinberg, Éva Tardos**: *Algorithm Design*, 1st Edition, Pearson, 2005.
3. **Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani**: *Algorithms*, 1st Edition, McGraw-Hill, 2008.
4. **Steven S. Skiena**: *The Algorithm Design Manual*, 3rd Edition, Springer, 2020.
5. **Robert Sedgewick, Kevin Wayne**: *Algorithms*, 4th Edition, Addison-Wesley, 2011.

E-Resources:

1. https://nptel.ac.in/courses/106/101/106101060/
2. http://cse01-iiith.vlabs.ac.in/
3. http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=IntroToAlgorithms
4. https://www.coursera.org/specializations/algorithms

COURSE OBJECTIVES:

The course objectives for the Algorithms syllabus aim to provide students with a comprehensive understanding of fundamental algorithmic concepts and their applications in problem-solving. Students will analyze the efficiency of algorithms using various performance metrics, including asymptotic notations, and explore algorithmic strategies such as brute force, divide and conquer, and dynamic programming. The course will cover advanced topics, including greedy techniques, backtracking, and branch-and-bound methods, while investigating the limitations of algorithmic power through discussions on NP-completeness and approximation algorithms. Additionally, students will develop proficiency in designing and implementing algorithms to solve complex problems effectively. Ultimately, the course seeks to equip students with the skills necessary for analyzing, designing, and optimizing algorithms in various computational contexts.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Analyze the non-recursive and recursive algorithms and represent the efficiency of these algorithms in terms of the standard asymptotic notations.
CO2	Acquire the knowledge of brute force and divide and conquer techniques to design algorithms and apply these methods in solving a given problem.
CO3	Master the decrease and conquer, transform and conquer algorithm design techniques, and understand time versus space trade-offs.
CO4	Learn greedy method and dynamic programming methods, applying these methods to design algorithms that solve given problems. Understand the importance of backtracking and branch-and-bound algorithm design techniques to solve a given problem.
CO5	Evaluate and analyze the complexity of algorithms and understand the classifications of problems, including P, NP, and NP-complete problems.

TEACHING PEDOGOGY

The course will utilize a mix of lecture-based instruction and hands-on lab sessions to foster a comprehensive understanding of algorithms and their applications. Lectures will cover theoretical aspects of algorithmic problem-solving, efficiency, and different algorithmic techniques such as brute force, divide and conquer, and dynamic programming. Lab sessions will focus on implementing these algorithms through programming exercises, enabling students to gain practical experience. Group discussions and case studies will enhance collaborative learning and critical thinking about algorithmic approaches and their real-world implications**.**

SKILL DEVELOPMENT

1. Proficiency in analyzing the efficiency of algorithms using asymptotic notation.
2. Ability to implement and analyze various algorithmic techniques, including sorting and searching algorithms.
3. Experience in solving complex problems using divide and conquer and dynamic programming methods.
4. Understanding of advanced topics such as NP-completeness and approximation algorithms.
5. Development of algorithm design skills through practical coding assignments and projects.

24MCA25: ARTIFICIAL INTELLIGENCE

Course Code:	24MCA25	**Course Title**	Artificial Intelligence
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	4	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**Introduction to Al Problem solving 15 Hours** Introduction to Al Problem solving: Problem-solving agents; Uninformed search strategies: DFS, BFS; Informed Search: Best First Search, A* search, AO* search; Minimax Search, Alpha-Beta pruning. Knowledge-based Agents, The Wumpus world as an example world, Logic, Propositional logic, First-order predicate logic, Propositional versus first-order inference, Unification and lifting, Forward chaining, Backward chaining, Resolution, Truth maintenance systems		
II	**Planning and Fuzzy Logic 15 HOURS** Planning – Representation of planning – Partial order planning –Planning and acting in real world – Acting under uncertainty – Bayes's rules – Semantics of Belief networks – Inference in Belief networks – Making simple decisions – Making complex decisions. Uncertainties: Non-monotonic reasoning, Probabilistic reasoning, Fuzzy logic: Theory of Fuzzy sets, Operations on Fuzzy sets and Fuzzy logic, Reasoning with Fuzzy logic;		
III	**Weak slot and Filler Structure 15 HOURS** Semantic Nets, Frames. Strong slot Filler Structures: Conceptual Dependency, scripts. AI Programming Languages (PROLOG): Introduction, How Prolog works, Backtracking, CUT and FAIL operators, Built –in Goals, Lists, Search in Prolog.Connectionist Models / ANN: Foundations for Connectionist Networks, Biological Inspiration; Different Architectures and output functions: Feed forward, Feedback, Recurrent Networks, step, Sigmoid and different functions.		
IV	**Language Models 15 HOURS** AI applications – Language Models – Information Retrieval- Information Extraction –Fields of Natural Language Processing, Chatbots and its types, Artificially Intelligent Chatbots, Introduction to Chatbot Applications (Retrieval based- Conversation based)- Deploy a chatbot using TensorFlow in python. Machine Translation – Speech Recognition – Robot – Hardware – Perception – Planning – Moving.		

REFERENCE BOOKS:

1. Russell, S. and Norvig, P., **“Artificial Intelligence - A Modern Approach”**, 4th Edition, Pearson, 2020.
2. Nilsson, Nils J., **“The Quest for Artificial Intelligence”**, 1st Edition, Cambridge University Press, 2010, ISBN: 978-0-521-12293-1.
3. John J. Craig, **“Introduction to Robotics: Mechanics and Control”**, 4th Edition, Pearson, 2018.
4. Elaine Rich, Kevin Knight, Shivashankar B. Nair, **“Artificial Intelligence”**, 3rd Edition, McGraw-Hill, 2008.
5. N.P. Padhy, **“Artificial Intelligence and Intelligent Systems”**, 1st Edition, Oxford Higher Education, Oxford University Press, 2005.
6. George F. Luger, **“Artificial Intelligence: Structures and Strategies for Complex Problem Solving”**, 6th Edition, Pearson Education, 2008.
7. Ivan Bratko, **“PROLOG Programming for Artificial Intelligence”**, 4th Edition, Pearson, 2011.

COURSE OBJECTIVES:

The course objectives for the Artificial Intelligence syllabus aim to provide students with a fundamental understanding of problem-solving techniques and the development of intelligent agents. Students will learn to analyze and implement search strategies, including uninformed and informed search methods, as well as knowledge-based systems and their applications. The course will explore planning techniques and the role of uncertainty in AI decision-making, alongside the fundamentals of robotics and computer vision. Additionally, students will gain hands-on experience in developing AI applications, such as chatbots and machine translation systems, equipping them with the skills to apply AI technologies effectively in real-world scenarios.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Understand problem-solving techniques in AI.
CO2	Analyze and apply knowledge-based systems.
CO3	Design and implement search strategies in AI.
CO4	Apply logical systems for AI problem solving and chatbot communication.
CO5	Evaluate AI applications in various fields, including natural language processing, Language models.

TEACHING PEDOGOGY

The course will utilize a combination of lecture-based instruction and hands-on lab sessions to engage students in the practical applications of AI concepts. Interactive discussions will encourage collaborative learning, while case studies will provide real-world context to theoretical principles. Students will participate in coding exercises to develop AI applications, employing tools like TensorFlow for chatbot deployment and machine learning tasks. Regular assessments and group projects will reinforce learning outcomes and foster teamwork.

SKILL DEVELOPMENT

1. Proficiency in problem-solving techniques and algorithm design in AI.
2. Hands-on experience in implementing search strategies and knowledge-based systems.
3. Understanding and application of planning and decision-making under uncertainty.
4. Familiarity with robotics fundamentals and computer vision techniques.
5. Development skills in AI applications, including chatbots and natural language processing solutions.

24MCA26: EMPLOYABILITY AND SKILL DEVELOPMENT

Course Code:	24MCA26	**Course Title**	Employability and Skill Development
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	2	**Domain**	COMPUTER SCIENCE
Syllabus			
I	**Soft Skills: Communication Skills** **15 HOURS** Master verbal and non-verbal communication, enhance oral and written communication, develop effective listening skills, and improve presentation skills.**Interpersonal Skills**: Learn the importance of teamwork, conflict resolution strategies, and building positive relationships with team members. **Leadership Skills**: Understand the significance of leadership and develop effective leadership qualities. **Practical Exercise**: As a team leader, draft an appreciation letter to team members for successful project completion		
II	**Quantitative Aptitude, Logical Reasoning, and Analytical Ability** **15 HOURS** **Quantitative Aptitude**: Practice percentage calculations, profit and loss, and other basic numerical problems. **Logical Reasoning**: Solve problems related to coding and decoding, blood relations, and non-verbal reasoning. **Analytical Ability**: Work on statement and assumptions, and data interpretation problems. **Practical Exercise**: Conduct a mock competitive exam covering quantitative aptitude, logical reasoning, and analytical ability		
III	**Career Development and Workplace Etiquette** **15 HOURS** **Career Development**: Perform a SWOC analysis for self-assessment, set career goals, create a career plan, and develop job search strategies. Learn resume preparation, including different types of resumes and effective writing tips. **Workplace Etiquette**: Understand time management, dress code, personal grooming, office manners, and meeting etiquette. Learn about professional ethics and their features. **Practical Exercise**: Prepare a resume with at least two references and conduct a mock interview based on the resume		
IV	**Interview Skills and Professional Networking:** **15 HOURS** **Interview Skills**: Explore different types of interviews, basic interview skills, stages of an interview, and parameters for scoring. Learn how to handle rejections and failures. **Group Discussions**: Understand the steps and strategies for effective group discussions. **Professional Networking**: Learn the meaning, importance, and methods of professional networking. **Practical Exercise**: Conduct mock group discussions and interviews.		

REFERENCE BOOKS:

1. Barun K Mitra, Personality Development and Soft Skills, Oxford university press, NewDelhi.
2. Gitangshu Adhikary, Communication and Corporate Etiquette, Notion Press, Mumbai.
3. Seema Gupta, Soft Skills- Interpersonal & Intrapersonal skills development, V&SPublishers, New Delhi.
4. Dr. R S Aggarwal, Quantitative Aptitude, S.Chand Publication, New Delhi.
5. Bittu Kumar, Mastering MS Office, V&S Publisher, New Delhi

COURSE OBJECTIVES:

The course objectives for the Professional Skills Development syllabus aim to equip students with essential soft skills required for effective communication, career development, and problem-solving. Students will master both verbal and non-verbal communication techniques, enhance their writing and presentation skills, and develop active listening abilities. The course will also cover quantitative aptitude, logical reasoning, and analytical skills to prepare students for competitive examinations and real-world problem-solving. Additionally, students will learn the nuances of career development, including resume writing, time management, and workplace etiquette. The course further aims to develop effective interview skills, group discussion strategies, and the importance of professional networking.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Master both verbal and non-verbal communication, enhance writing and presentation abilities, and improve active listening skills.
CO2	Apply quantitative aptitude, logical reasoning, and analytical skills to solve practical problems and perform well in competitive examinations.
CO3	Create a professional resume, understand career development strategies, and practice workplace etiquette and time management.
CO4	Develop effective interview skills, participate confidently in group discussions,

	and understand the importance of professional networking.
CO5	Build leadership qualities and interpersonal skills for teamwork, conflict resolution, and positive relationship-building within a professional environment.

TEACHING PEDOGOGY

The teaching pedagogy for this course will involve a mix of lectures, interactive sessions, and practical exercises. Emphasis will be placed on active participation through role-playing, group activities, and mock interviews. Case studies and real-life scenarios will be discussed to provide context to soft skills like leadership, communication, and workplace etiquette.

SKILL DEVELOPMENT

1. Students will enhance verbal, non-verbal, and written communication skills, critical for workplace success.
2. Proficiency in solving numerical problems, logical puzzles, and data interpretation, sharpening analytical thinking.
3. Expertise in creating professional resumes, developing career plans, and practicing essential workplace etiquette.
4. Students will gain confidence in interviews, group discussions, and building professional networks.
5. Building leadership qualities and teamwork skills through real-life simulations, fostering effective collaboration and conflict resolution.

24MCA27: JAVA PROGRAMMING LAB

Course Code:	24MCA27P	**Course Title**	Java Programming Lab
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	2	**Domain**	COMPUTER SCIENCE
Syllabus			

LIST OF JAVA PROGRAMMING LAB PROGRAMS

1. Program to illustrate class, objects and constructors
2. Program to implement overloading, overriding, polymorphism etc.
3. Program to implement the usage of packages
4. Program to create user defined and predefined exception
5. Program for handling file operation
6. Directory manipulation in java
7. Implement the concept of multithreading and synchronization
8. Program to implement Generic class and generic methods
9. Socket programming to implement communications
10. Broadcasting program using UDP protocol
11. Program for downloading web pages from the internet using URL
12. Program to implement JDBC in GUI and Console Application
13. Applet program for passing parameters
14. Applet program for loading an image and running an audio file
15. Program for event-driven paradigm in Java
16. Event driven program for Graphical Drawing Application
17. Program that uses Menu driven Application

COURSE OBJECTIVES:

The course objectives for the Java Programming Lab aim to provide students with a practical understanding of object-oriented programming principles, focusing on classes, objects, and constructors. Students will learn to implement various programming concepts such as method overloading, overriding, and polymorphism. The course will cover exception handling, file operations, and multithreading to equip students with skills to manage errors and perform concurrent programming. Additionally, students will gain experience in socket programming and JDBC for database interactions, along with developing GUI applications using applets and event-driven programming.

COURSE OUTCOME	
CO CODE	**COURSE DESCRIPTION**
CO1	Understand the principles of object-oriented programming by illustrating classes, objects, and constructors.
CO2	Demonstrate the ability to implement method overloading, overriding, and polymorphism in Java applications.
CO3	Acquire skills in exception handling and file operations, including directory manipulation.
CO4	Implement multithreading and synchronization in Java programs to handle concurrent tasks effectively.
CO5	Develop applications using socket programming and JDBC, and create graphical user interfaces using applets and event-driven programming techniques.

TEACHING PEDOGOGY

The teaching pedagogy for the Java Programming Lab will include a combination of lectures and hands-on programming sessions. Students will be introduced to fundamental Java concepts through theoretical explanations followed by practical demonstrations. Lab sessions will involve guided exercises where students will write and debug code, fostering an interactive learning environment. Group projects and pair programming activities will encourage collaboration and peer learning. Additionally, case studies of real-world applications will be used to illustrate the practical applications of Java programming concepts

SKILL DEVELOPMENT

- Students will master key concepts such as classes, objects, inheritance, and polymorphism.
- Skills in creating and managing user-defined and predefined exceptions will be developed.
- Understanding of concurrency in Java applications through the implementation of multithreading techniques.
- Ability to design and implement socket-based communication and UDP broadcasting in Java.
- Practical experience in using JDBC to connect Java applications with databases and executing SQL queries.

24MCA28: ARTIFICIAL INTELLIGENCE LAB USING PYTHON

Course Code:	24MCA28P	**Course Title**	Artificial Intelligence Lab Using Python
Course Type	DSC	**Contact Hours**	4 Hours per Week **Total**:60 Hours
Credit	2	**Domain**	COMPUTER SCIENCE
Syllabus			

LIST OF PROGRAMS OF ARTIFICIAL INTELLIGENCE LAB USING PYTHON

1. Implementation of Depth First Search (DFS) Algorithm.
2. Implementation of Breadth First Search (BFS) Algorithm.
3. Best First Search Algorithm Implementation.
4. *A Search Algorithm for Pathfinding.**
5. *AO Search Algorithm for Problem Solving.**
6. Minimax Algorithm for Game Playing.
7. Alpha-Beta Pruning to Optimize Minimax Algorithm.
8. Knowledge Representation using Propositional Logic in Prolog.
9. First-Order Predicate Logic (FOPL) in Prolog for Reasoning.
10. Resolution Algorithm in Propositional Logic.
11. Implementation of Forward Chaining in Prolog.
12. Implementation of Backward Chaining in Prolog.
13. Simple Bayesian Network Representation for Inference.
14. Fuzzy Logic: Implementation of Fuzzy Set Operations.
15. Partial Order Planning Algorithm for Task Scheduling.
16. Prolog Programming with Backtracking, CUT, and FAIL.
17. ANN Model: Simple Feedforward Neural Network using Python.
18. Chatbot Development using TensorFlow in Python.
19. Speech Recognition System using Python and Libraries (e.g., Speech Recognition).
20. Simple Machine Translation System using Python.

COURSE OBJECTIVES:

The course objectives for the AI Lab using pythonto provide students with hands-on experience in implementing various algorithms for problem-solving in artificial intelligence. Students will learn to apply search techniques such as Breadth-First Search and A* search for optimal pathfinding. The lab will also cover game theory concepts through the implementation of Minimax search for two-player games and the solution of constraint satisfaction problems like the 4-Queens problem. Additionally, students will gain practical skills in image processing using the OpenCV library, machine learning tasks such as classification and clustering, and natural language processing (NLP) techniques using Python. The course ultimately seeks to enhance students' abilities to develop AI applications and perform data analysis.

COURSE OUTCOME	
COURSE CODE	**COURSE DESCRIPTION**
CO1	Implement search algorithms, including Breadth-First Search and A*, to solve complex problems like Tic-Tac-Toe and optimal pathfinding.
CO2	Develop solutions for constraint satisfaction problems, such as the 4-Queens problem, and apply Minimax search for two-player games.
CO3	Utilize the OpenCV library for image processing tasks, including resizing, blurring, edge detection, and segmentation.
CO4	Apply machine learning algorithms such as Decision Trees, Naïve Bayes, and K-Means clustering to classify and analyze datasets.
CO5	Implement natural language processing tasks using Python NLTK, and create a chatbot using Python, showcasing a range of AI applications.

TEACHING PEDOGOGY

The teaching pedagogy for the AI and Python Lab will combine theoretical lectures with practical programming sessions. Students will be introduced to various algorithms through conceptual explanations, followed by guided coding exercises in a lab environment. Hands-on projects will allow students to implement algorithms for search, image processing, and machine learning. Collaborative group work will be encouraged to foster peer learning and enhance problem-solving skills. Additionally, real-world case studies will be presented to demonstrate the applications of AI and machine learning techniques

SKILL DEVELOPMENT

1. Students will develop skills in coding search algorithms and problem-solving techniques relevant to AI.
2. Proficiency in using OpenCV for various image manipulation and analysis tasks will be cultivated.
3. Understanding of different classification and clustering algorithms, along with practical experience in data analysis.
4. Capability to perform key NLP tasks such as tokenization, stemming, and named entity recognition using Python.
5. Hands-on experience in developing AI-based conversational agents will enhance programming and application design skills.