UNIVERSITY OF KALYANI

UG 4 YEAR

Computer Science

(HONOURS/ HONOURS WITH RESEARCH)
SYLLABUS

(Under NEP 2020)
W.E.F. the Academic Session 2023-24

SEMESTER I							
Course Code	**Course title**	**Nature of Course**	**Credit of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-1	Computer Fundamentals and Programming using C	Major (Theory)	4	4	15	60	75
CS-MJ-P-1	Programming using C Lab	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-MI-T-1	Computer Fundamentals and Programming using C	Minor (Theory)	3	3	10	25	35
CS-MI-P-1	Programming using C Lab	Minor (Practical)	1	2	Problem(10)+ Viva(5):15		15
CS-MU-T-1	Computer Science for Beginners	Multidisciplinary Course	3	3	10	35	45
CS-SEC-P-1	Office Automation Lab	Skill Enhancement Course	3	3	Problem: 35, Viva: 5, Lab Notebook: 5		45
		Value Added Course	4	4	10	40	50
Total			20				

SEMESTER II							
Course Code	**Course title**	**Nature of Course**	**Credit of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-2	Digital System Design	Major (Theory)	4	4	15	60	75
CS-MJ-P-2	Digital System Design Lab	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-MI-T-2	Computer Fundamentals and Programming using C	Minor (Theory)	3	3	10	25	35
CS-MI-P-2	Programming using C Lab	Minor (Practical)	1	2	Problem(10)+ Viva(5):15		15
CS-MU-P-2	Office Automation	Multidisciplinary Course	3	3	10	35	45
AECC-1		Ability Enhancement Course	4	4	10	40	50
CS-SEC-P- 2	Web Development and Applications Lab	Skill Enhancement Course	3	3	Problem: 35, Viva: 5, Lab Notebook: 5		45
CS-SI-1	Summer Internship	Additional for Certificate Diploma	2		10	40	50
Total			20/22				

SEMESTER III							
Course Code	**Course title**	**Nature of Course**	**Credit of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-3	Computer Organization & Architecture	Major (Theory)	6	6	15	60	75
CS-MI-T-3	Database Management Systems	Minor (Theory)	3	3	10	25	35
CS-MI-P-3	Database Management Systems Lab	Minor (Practical)	1	2	Problem(10)+ Viva(5):15		15
CS-MU-T-3	AI for Everyone	Multidisciplinary Course	3	3	10	35	45
CS-SEC-P-3	Data Analysis through Python/R	Skill Enhancement Course	3	3	Problem: 35, Viva: 5, Lab Notebook: 5		45
		Value Added Course	4	4	10	40	50
Total			20				265

SEMESTER IV							
Course Code	**Course title**	**Nature of Course**	**Credit of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-4	Data Structures	Major (Theory)	4	4	15	60	75
CS-MJ-P-4	Data Structures Lab	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-MJ-T-5	Discrete Mathematics	Major (Theory)	6	6	15	60	75
CS-MI-T-4	Database Management Systems	Minor (Theory)	3	3	10	25	35
CS-MI-P-4	Database Management Systems Lab	Minor (Practical)	1	2	Problem(10)+ Viva(5):15		15
AECC-2		Ability Enhancement Course	4	3			
CS-SI-2	Summer Internship	Additional for Certificate Diploma	4		10	40	50
Total			20/24				

SEMESTER V							
Course Code	**Course title**	**Nature Of Course**	**Credit of Course**	**Class hour/ week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-6	Database Management Systems	Major (Theory)	4	4	15	60	75
CS-MJ-P-6	Database Management Systems Lab	Major (practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-MJ-T-7	Design and Analysis of Algorithms	Major (Theory)	6	6	15	60	75
CS-MJ-T-8	Microprocessor and its applications	Major (Theory)	4	3	10	25	35
CS-MJ-P-8	Microprocessor and Its Application Lab	Major (practical)	2	2	Problem(10)+ Viva(5):15		15
CS-MI-P-5	Programming In Python	Minor (Practical)	4	4	Problem: 35, Viva: 10, Lab Notebook: 5		50
Total			20				

N.B.: Any one of the following (CS-MJ-T-6 or CS-MJ-T-8) to be taught Decided by the concerned College.

SEMESTER VI							
Course Code	**Course title**	**Nature of Course**	**Credit of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-9	Data Communication and Computer Networks	Major (Theory)	6	6	15	60	75
CS-MJ-T-10	Operating Systems	Major (Theory)	6	6	15	60	75
CS-MJ-T-11	Object Oriented Programming using JAVA/C++*	Major (Theory)	4	4	15	60	75
CS-MJ-P-11	Object Oriented Programming using JAVA/C++ Lab*	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-SI-3	Summer Internship		2		10	40	50
Total			20				

*** Either JAVA or C++ will be taught.**

SEMESTER VII							
Course Code	**Course title**	**Nature of Course**	**Credit Of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-12	Advanced Mathematics	Major (Theory)	6	6	15	60	75
CS-MJ-T-13	Artificial Intelligence	Major (Theory)	4	4	15	60	75
CS-MJ-P-13	Artificial Intelligence Lab	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-MJ-T-14	Theory of Computation	Major (Theory)	6	6	15	60	75
CS-MJ-T-15	Software Engineering	Major (Theory)	6	4	15	35	50
CS-MI-P-15	Web Development and Applications	Minor (Practical)	4	4	Problem:35, Viva: 10, Lab Notebook: 5		50
			28				

N.B.: Any one of the following (CS-MJ-T-14 or CS-MJ-T-15) to be taught Decided by the concerned College.

SEMESTER VIII							
Course Code	**Course title**	**Nature of Course**	**Credit of Course**	**Class hour/week**	**Evaluation**		**Total**
					Internal	**Semester End**	
CS-MJ-T-16	Machine Learning	Major (Theory)	4	4	15	60	75
CS-MJ-P-16	Machine Learning Lab	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
CS-MJ-T-17	Soft Computing	Major (Theory)	6	6	15	60	75
CS-MJ-T-18	Data Warehouse and Data Mining	Major (Theory)	4	4	15	60	75
CS-MJ-P-18	Data Warehouse and Data Mining Lab	Major (Practical)	2	3	Problem: 60, Viva: 10, Lab Notebook: 5		75
For B.Sc. Honours							
CS-MJ-T-19	Computer Graphics	Major (Theory)	6	4	15	60	75
CS-PRO-HON	Project	Major	6			75	75
Total			30				
For B.Sc. Honours with Research							
CS-PRO-REC		Research Project / Dissertation [Honours With Research]	12				150
Total			24				

Detail Syllabus of Semester-I

CS-MJ-T-1	**Computer Fundamentals and Programming using C**	Semester	Credit	Hours	Evaluation
Major		I	4	40	15+60
Theory					

Course Objectives

This course aims to develop students' foundational understanding of computers, number systems, Boolean algebra, and problem-solving techniques, while imparting programming skills in C. Through hands-on practice, students will learn to design, implement, and debug efficient programs involving data structures, pointers, file handling, and basic algorithms, preparing them for advanced computing and application development.

Syllabus

Unit I	**Introduction to Computer and Problem Solving** Information and Data. Hardware: CPU, Primary and Secondary storage, I/O devices Software: Systems and Application.	**3 Hrs.**
Unit II	**Introduction to Programming Languages** Machine Language, Assembly Language, High Level Language. Problem Solving: Flow Charts, Decision Tables and Pseudo codes.	**3 Hrs.**
Unit III	**Number Representation** Weighted Codes, Non-weighted codes, Positional, Binary, Octal, Hexadecimal, Binary Coded Decimal (BCD), Conversion of bases. Complement notions. Binary Arithmetic, Binary Codes: Gray, Alphanumeric, ASCII, EBCDIC; Parity Bits. Single Error-Detecting and Correcting Codes, Hamming Codes, Fixed and Floating Point Arithmetic.	**4 Hrs.**
Unit IV	**Boolean Algebra** Fundamentals of Boolean Algebra, Switches and Inverters, Functionally Complete Gates (AND, OR, NOT), NAND, NOR. Switching function and Boolean function. De Morgan's theorem, Minterm and Maxterm, Truth table and minimization of switching function up to four variables, Algebraic and K-map method of logic circuit synthesis: Two level and Multi level.	**4 Hrs.**
Unit V	**Introduction to C** C character set, Identifiers and keywords, Data types, Declarations, Expressions, statements and symbolic constants. Input-Output - getchar, putchar, scanf, printf, gets, puts, functions. Pre-processor commands - #include, #define, #ifdef	**6 Hrs.**
Unit VI	**Operators, expressions and storage Types** Arithmetic, unary, logical, bit-wise, assignment and conditional operators. Storage types - Automatic, external, register and static variables.	**3 Hrs.**
Unit VII	**Functions** Defining and accessing, passing arguments, Function prototypes, Recursion, Library functions, Static functions.	**3 Hrs.**
Unit VIII	**Arrays and Strings** Arrays- Defining and processing, Passing arrays to a function, Multi dimensional arrays. Strings- Defining and operations on strings.	**4 Hrs.**
Unit IX	**Pointers and Structures** Pointers- Declarations, Passing pointers to a function, Operations on pointers, Pointer Arithmetic, Pointers and arrays, Arrays of pointers function pointers. Structures- Defining and processing, Passing to a function, Unions, typedef, array of structure, and pointer to structure.	**4 Hrs.**
Unit X	**File Handling** Definitions, concept of record, file operations: Storing, creating, retrieving, updating Sequential, relative, indexed and random access mode, Files with binary mode(Low level), performance of Sequential Files, Direct mapping techniques: Absolute, relative and indexed sequential files (ISAM) concept of index, levels of index, overflow of handling. File operation: creation, copy, delete, update, text file, binary file.	**6 Hrs.**

Recommended Books

1. P. K. Sinha & Priti Sinha , —Computer Fundamentalsll, BPB Publications, 2007.
2. Dr. Anita Goel, Computer Fundamentals, Pearson Education, 2010.
3. Kernighan, Brian W., and Dennis M. Ritchie.The C programming language. 2006.
4. Schildt, Herbert, and C. Turbo. "C: the complete reference, Osborne." (2000).
5. Balagurusamy, E. programming in ANSI C. Tata McGraw-Hill Education, 2002.
6. Kanetkar, Yashavant P. Let us C. BPB publications, 2016.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- **Remember** – Recall fundamental concepts of computer hardware, software, programming languages, number systems, Boolean algebra, and C language syntax.
- **Understand** – Explain problem-solving techniques using flowcharts, decision tables, pseudo codes, and describe data representation and logic circuit simplification methods.
- **Apply** – Implement algorithms in C using functions, arrays, pointers, structures, and file handling to solve basic computational problems.
- **Analyze** – Differentiate between various storage types, operators, and programming constructs; debug and optimize C programs.
- **Evaluate** – Assess the efficiency of algorithms and file-handling techniques for different problem scenarios.

CS-MJ-P-1	**Programming using C Lab**	Semester	Credit	Hours	Evaluation
Major		I	2	30	60+10+5
Practical					

Course Objectives

This course aims to build a strong foundation in computer fundamentals, problem-solving, and C programming, covering number systems, Boolean algebra, and core programming constructs such as functions, arrays, pointers, structures, and file handling, enabling students to design and implement efficient solutions to real-world problems.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

Introduction to Computer and Problem Solving

(No direct coding experiments — focus on problem solving & logic design exercises in the lab before programming)

Draw **flowcharts** and write **pseudocode** for:

- Finding the area and circumference of a circle.
- Finding the largest of three numbers.
- Checking whether a number is prime.
- Demonstrate simple algorithms using **decision tables**.

Introduction to C programming, Number Representation & Boolean Algebra

1. Write a C program to convert a number from decimal to binary, octal, and hexadecimal.
2. Write a C program to convert binary to decimal.
3. Write a C program to perform binary addition, subtraction, multiplication, and division.
4. Write a C program to find the **1's and 2's complement** of a binary number.
5. Write a C program to find the **Gray code** of a given binary number and vice versa.
6. Write a C program to read the radius of a circle and find its area and circumference.
7. Write a C program to read three numbers and find the largest among them.
8. Write a C program to check whether a number is prime.
9. Write a C program to generate the first *n* prime numbers.
10. Write a C program to read a number, find the sum of its digits, reverse the number, and check whether it is a palindrome.
11. Write a C program to read numbers continuously until the user enters 999, and find the sum of only positive numbers.
12. Write a C program to read percentage of marks and display an appropriate message (else-if ladder).
13. Write a C program to find the roots of a quadratic equation (switch-case statement).

Operators, Expressions & Storage Types

- Modify earlier programs to use:
- Logical, bitwise, and conditional operators.
- Static and register variables to observe lifetime and scope.

Functions

1. Write a C program to check whether a number is prime by defining an isprime() function.
2. Write a C program to find factorial using recursion.
3. Write a C program to find GCD and LCM using functions.
4. Write a C program to read, display, and find the trace of a square matrix.
5. Write a C program to read, display, and add two $m \times n$ matrices using functions.
6. Write a C program to read, display, and multiply two matrices using functions.

Arrays and Strings

1. Write a C program to read marks scored by *n* students and find the average.
2. Write a C program to remove duplicate elements from a single-dimensional array.
3. Write a C program to demonstrate string functions in C.
4. Write a C program to read a string and count alphabets, digits, vowels, consonants, spaces, and special characters.

Pointers and Structures

1. Write a C program to demonstrate pointers in C.
2. Write a C program to reverse a string using pointers.
3. Write a C program to swap two numbers using pointers.
4. Write a C program to demonstrate a student structure to read and display records of *n* students.
5. Write a C program to demonstrate the difference between a structure and a union.

File Handling

1. Write a C program to create and write into a text file.
2. Write a C program to read from a text file and count words and lines.
3. Write a C program to copy a file to another file.
4. Write a C program to store and retrieve, update and delete records in a binary file.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design algorithms using flowcharts and pseudocode for basic computational problems.
- Write and debug C programs using core concepts like functions, arrays, pointers, structures, and file.
- Apply programming to solve mathematical, logical, and data processing tasks.
- Demonstrate practical proficiency in C through hands-on lab exercises.

CS-MI-T-1	**Computer Fundamentals and Programming using C**	Semester	Credit	Hours	Evaluation
Minor		I	3	30	10+25
Theory					

Course Objectives

This course aims to provide students with a solid foundation in computer fundamentals, data representation, Boolean algebra, and problem-solving techniques, along with an introduction to the C programming language for developing efficient algorithms and implementing structured solutions to computational problems.

Syllabus

Unit I	**Introduction to Computer and Problem Solving** Information and Data. Hardware: CPU, Primary and Secondary storage, I/O devices Software: Systems and Application.	**3 Hrs.**
Unit II	**Introduction to Programming Languages** Machine Language, Assembly Language, High Level Language. Problem Solving: Flow Charts, Decision Tables and Pseudo codes.	**3 Hrs.**
Unit III	**Number Representation** Weighted Codes, Non-weighted codes, Positional, Binary, Octal, Hexadecimal, Binary Coded Decimal (BCD), Conversion of bases. Complement notions. Binary Arithmetic, Binary Codes: Gray, Alphanumeric, ASCII, EBCDIC; Parity Bits. Single Error-Detecting and Correcting Codes, Hamming Codes, Fixed and Floating Point Arithmetic.	**4 Hrs.**
Unit IV	**Boolean Algebra** Fundamentals of Boolean Algebra, Switches and Inverters, Functionally Complete Gates (AND, OR, NOT), NAND, NOR. Switching function and Boolean function. De Morgan's theorem, Minterm and Maxterm, Truth table and minimization of switching function up to four variables, Algebraic and K-map method of logic circuit synthesis: Two level and Multi level.	**4 Hrs.**
Unit V	**Introduction to C** C character set, Identifiers and keywords, Data types, Declarations, Expressions, statements and symbolic constants. Input-Output - getchar, putchar, scanf, printf, gets, puts, functions. Pre-processor commands - #include, #define, #ifdef	**6 Hrs.**
Unit VI	**Operators, expressions and storage Types** Arithmetic, unary, logical, bit-wise, assignment and conditional operators. Storage types - Automatic, external, register and static variables.	**3 Hrs.**
Unit VII	**Functions** Defining and accessing, passing arguments, Function prototypes, Recursion, Library functions, Static functions.	**3 Hrs.**
Unit VIII	**Arrays and Strings** Arrays- Defining and processing, Passing arrays to a function, Multi dimensional arrays. Strings- Defining and operations on strings.	**4 Hrs.**

Recommended Books

1. P. K. Sinha & Priti Sinha , —Computer Fundamentalsǁ, BPB Publications, 2007.
2. Dr. Anita Goel, Computer Fundamentals, Pearson Education, 2010.
3. Kernighan, Brian W., and Dennis M. Ritchie.The C programming language. 2006.
4. Schildt, Herbert, and C. Turbo. "C: the complete reference, Osborne." (2000).
5. Balagurusamy, E. programming in ANSI C. Tata McGraw-Hill Education, 2002.
6. Kanetkar, Yashavant P. Let us C. BPB publications, 2016.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember – Recall the basic concepts of computer hardware, software, number systems, Boolean algebra, and C language syntax.
- Understand – Explain problem-solving techniques using flowcharts, decision tables, pseudo codes, and describe data representation and logic simplification methods.
- Apply – Implement algorithms in C using variables, operators, functions, arrays, and strings to solve computational problems.
- Analyze – Differentiate between storage types, operators, and programming constructs; simplify logic expressions using Boolean algebra and Karnaugh maps.
- Evaluate – Assess the efficiency and correctness of algorithms and C programs through debugging and testing.

CS-MJ-P-1	**Programming using C Lab**	Semester	Credit	Hours	Evaluation
Minor		I	1	30	15
Practical					

Course Objectives

This course aims to build a strong foundation in computer fundamentals, problem-solving, and C programming, covering number systems, Boolean algebra, and core programming constructs such as functions, arrays and pointers enabling students to design and implement efficient solutions to real-world problems.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

Introduction to Computer and Problem Solving

(No direct coding experiments — focus on problem solving & logic design exercises in the lab before programming)

Draw **flowcharts** and write **pseudocode** for:

- Finding the area and circumference of a circle.
- Finding the largest of three numbers.
- Checking whether a number is prime.
- Demonstrate simple algorithms using **decision tables**.

Introduction to C programming, Number Representation & Boolean Algebra

1. Write a C program to convert a number from decimal to binary, octal, and hexadecimal.
2. Write a C program to convert binary to decimal.
3. Write a C program to perform binary addition, subtraction, multiplication, and division.
4. Write a C program to find the **1's and 2's complement** of a binary number.
5. Write a C program to find the **Gray code** of a given binary number and vice versa.
6. Write a C program to read the radius of a circle and find its area and circumference.
7. Write a C program to read three numbers and find the largest among them.
8. Write a C program to check whether a number is prime.
9. Write a C program to generate the first *n* prime numbers.
10. Write a C program to read a number, find the sum of its digits, reverse the number, and check whether it is a palindrome.
11. Write a C program to read numbers continuously until the user enters 999, and find the sum of only positive numbers.
12. Write a C program to read percentage of marks and display an appropriate message (else-if ladder).
13. Write a C program to find the roots of a quadratic equation (switch-case statement).

Operators, Expressions & Storage Types

- Modify earlier programs to use:
- Logical, bitwise, and conditional operators.
- Static and register variables to observe lifetime and scope.

Functions

1. Write a C program to check whether a number is prime by defining an isprime() function.
2. Write a C program to find factorial using recursion.
3. Write a C program to find GCD and LCM using functions.
4. Write a C program to read, display, and find the trace of a square matrix.
5. Write a C program to read, display, and add two $m \times n$ matrices using functions.
6. Write a C program to read, display, and multiply two matrices using functions.

Arrays and Strings

1. Write a C program to read marks scored by *n* students and find the average.
2. Write a C program to remove duplicate elements from a single-dimensional array.
3. Write a C program to demonstrate string functions in C.
4. Write a C program to read a string and count alphabets, digits, vowels, consonants, spaces, and special characters.

Pointers and Structures

1. Write a C program to demonstrate pointers in C.
2. Write a C program to reverse a string using pointers.
3. Write a C program to swap two numbers using pointers.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design algorithms using flowcharts and pseudocode for basic computational problems.
- Write and debug C programs using core concepts like functions, arrays, pointers, structures, and file.
- Apply programming to solve mathematical, logical, and data processing tasks.
- Demonstrate practical proficiency in C through hands-on lab exercises.

CS-MU-T-1	**Computer Science for Beginners**	Semester	Credit	Hours	Evaluation
Multidisciplinary		I	3	30	10+35
Theory					

Course Objectives

To introduce students to the fundamental concepts of computers, including their history, hardware and software components, number systems, Boolean algebra, problem-solving techniques, programming languages, database systems, internet technologies, and information technology laws, enabling them to develop a foundational understanding necessary for further study in computer science and applications.

Syllabus

Unit I	**Generation of Computers** A brief history of generation of computers, Super, Mainframe, Mini and Personal Computer.	**1 Hrs.**
Unit II	**Introduction to Computer Hardware and Software** Components of modern computers: CPU, Primary and Secondary storage, I/O devices Software: Systems and Application.	**3 Hrs.**
Unit III	**Number Systems** A brief history of different number systems. Number representations and conversion rules in different number systems such as binary, octal, hexadecimal and decimal.	**2 Hrs.**
Unit IV	**Boolean Algebra** Fundamentals of Boolean Algebra, Switches and Inverters, Functionally Complete Gates (AND, OR, NOT), NAND, NOR. Switching function and Boolean function. De Morgan's theorem, Minterm and Maxterm, Truth table and minimization of switching function up to four variables, Algebraic and K-map method of logic circuit synthesis: Two level and Multi level.	**4 Hrs.**
Unit V	**Problem Solving** Flow Charts, Decision Tables and Pseudo codes.	**2 Hrs.**
Unit VI	**Programming Languages** A brief history of Programming languages: Machine Language, Assembly Language, High Leve l Language.	**3 Hrs.**
Unit VII	**Introduction to Database Management Systems** What is DBMS? Difference between DBMS and File structure, Architectures of DBMS. Different Types of DBMS.	**5 Hrs.**
Unit VIII	**Internet** History of the internet and different internet enabled services used in our daily life. Different internet service providers. Cloud services and service providers.	**4 Hrs.**
Unit IX	**Information Technology Laws** Information Technology laws provided to electronic commerce – electronic signatures, data protection, cyber security; penalties & offences under the IT Act, dispute resolution, and other contemporary issues.	**6 Hrs.**

Recommended Books and resources

1. P. K. Sinha & Priti Sinha , —Computer Fundamentalsll, BPB Publications, 2007.
2. Dr. Anita Goel, Computer Fundamentals, Pearson Education, 2010.
3. V. Rajaraman, Neeharika Adabala,- Fundamentals of Computers, PHI
4. IT Act 2000(INDIA).

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember – Recall the generations of computers, types of hardware, software, and number systems.
- Understand – Explain Boolean algebra concepts, problem-solving methods, and programming language classifications.
- Apply – Perform number system conversions, Boolean simplification, and design basic logic circuits.
- Analyze – Differentiate between DBMS and file systems, and compare internet-enabled services and cloud models.
- Evaluate – Assess the importance of IT laws in securing digital transactions and protecting data.
- Create – Develop basic algorithms and represent them using flowcharts, decision tables, and pseudocode.

CS-SEC-P-1	**Office Automation Lab**	Semester	Credit	Hours	Evaluation
Skill Enhancement		I	3	30	10+35
Practical					

Course Objectives

This skill development course aims to equip students with practical proficiency in office productivity tools, operating systems, and advanced spreadsheet-based data analysis, enabling them to create, manage, and present professional documents, presentations, and data-driven insights for real-world applications.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

Computer Software & OS Basics

1. **Installing and uninstalling software** – Demonstrate installation of a text editor, office suite, and uninstalling an application.
2. **Operating System Installation** – Install a Linux distribution (e.g., Ubuntu) or Windows in a virtual environment.
3. **Basic OS Navigation** – Using GUI and CUI commands for file/folder management.

Word Processing (MS Word / LibreOffice Writer)

4. **Creating & formatting documents** – Font, paragraph, and page settings.
5. **Working with tables** – Create a formatted timetable or invoice using tables.
6. **Mail Merge** – Generate personalized letters from a database file.
7. **Adding graphics and charts** – Insert and format images, shapes, and charts.
8. **Document protection** – Set passwords and restrict editing in a document.

Presentation (MS PowerPoint / LibreOffice Impress)

9. **Basic presentation creation** – Create a slide deck with text, bullet points, and images.
10. **Applying themes & animations** – Use slide transitions, custom animations, and multimedia.
11. **Interactive presentation** – Insert hyperlinks, action buttons, and navigation controls.
12. **Final presentation project** – Create a 5-slide presentation on a given topic.

Spreadsheet Basics (MS Excel / LibreOffice Calc)

13. **Worksheet creation & formatting** – Enter and format data for sales or student records.
14. **Formulas & functions** – Apply SUM, AVERAGE, COUNT, MIN, MAX, IF, AND, OR.
15. **Data cleaning** – Use TRIM, UPPER, LOWER, CONCAT, TEXTJOIN for formatting data.
16. **Sorting, filtering & conditional formatting** – Highlight top values or apply color scales.

Data Analysis & Advanced Spreadsheet Features

17. **Pivot Tables** – Create a pivot table to summarize sales data.
18. **Lookup functions** – Use VLOOKUP, HLOOKUP, and INDEX-MATCH for data retrieval.
19. **Statistical analysis** – Calculate mean, median, mode, variance, standard deviation.
20. **Regression & correlation** – Use built-in data analysis tools to analyze relationships.
21. **What-if analysis** – Apply Goal Seek and Data Tables for prediction models.
22. **Solver tool** – Solve an optimization problem (e.g., maximize profit).

Data Visualization & Reporting

23. **Basic chart creation** – Column, line, and pie charts.
24. **Advanced chart customization** – Combo charts, trendlines, sparklines, and data bars.
25. **Interactive dashboard** – Use slicers, form controls, and charts for a dynamic report.
26. **Monte Carlo simulation** – Generate random datasets and analyze results.
27. **Scenario analysis** – Create multiple business scenarios for decision-making.
28. **Final report & presentation** – Design a professional report with charts, pivot tables, and dashboards, and present it to the class.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Identify different categories of software, operating systems, and their installation procedures. (Remembering)
- Demonstrate the ability to create, edit, and format professional documents, presentations, and spreadsheets. (Applying)
- Apply formulas, functions, and logical operations for effective data manipulation and analysis. (Applying)
- Analyze datasets using pivot tables, charts, and advanced visualization techniques to derive meaningful insights. (Analyzing)
- Evaluate data integrity and apply validation, protection, and collaboration features in shared work environments. (Evaluating)
- Create interactive dashboards, scenario models, and professional reports for decision-making. (Creating)

Detail Syllabus of Semester-II

CS-MJ-T-2	**Digital System Design**	Semester	Credit	Hours	Evaluation
Major		II	4	40	15+60
Theory					

Course Objectives

To provide students with a comprehensive understanding of combinational and sequential digital circuits, data conversion techniques, and logic families, enabling them to design, analyze, and implement efficient digital systems for computational and control applications.

Syllabus

Unit I	**Combinational Circuits** Realization of AND and OR Gates using diodes and NOT Gate using transistors, Standard Gate Assemblies, IC chips packaging nomenclature, Half and Full Adder(3 bits), Multi-bit adders – Ripple carry and Carry Look Ahead Adder, Adder/subtractor, BCD-Adder, Data selectors/multiplexers – expansions, reductions, function realization, universal function realization, multi-function realization, Decoders/Demultiplexers : function realization, De-multiplexer and function realization, Encoder, Priority Encoder, Parity bit Generator/checker, Gray Code Generator, Code Converters, I/O features of BCD to 7- segment LED decoder/driver(7447/7448), Seven segment display unit, Comparators.	**15 Hrs.**
Unit II	**Sequential Circuits** Model of Sequential computing, Difference between Combinational and Sequential circuit, RS-Latch: using NAND and NOR Gates, RS Latch as a Static RAM Cell, Problems of Basic Latch circuits, Digital Clock – Duty Cycle, Rising time, Falling time, Clocked Flip Flops - SR, JK, D, T, Level Trigger and Edge Trigger, Excitation Functions of each flip-flops, Flip-flops with Preset and Clear, Application of Flip-flops: Asynchronous Counter (UP/DOWN) up to 4 bit counter, Mod – n Counter, Synchronous Counters – different mod counters, Ring counter, Registers: Registers with serial and parallel load, Shift Registers.	**15 Hrs.**
Unit III	**Data Converter** D/A Conversion principle using basic circuit, R-2R Ladder circuit, Counter based A/D converter, Successive approximation method for A/D conversion. DTL and TTL NAND gate circuits and its operations, Fan in & Fan out, Noise margin, SSI, MSI, LSI, and VLSI classifications.	**10 Hrs.**

Recommended Books

1. Digital Circuits, Combinational Circuit, Vol. 1 by D. Roy Choudhuri, Platinam Publication.
2. Digital Circuits, Sequential Circuit, Vol. 2 by D. Roy Choudhuri, Platinam Publication.
3. Digital Logic and Computer Design by M.Morris Mano, PHI
4. Digital Principle and Applications by Malvino & Leach, TMH
5. Digital Systems Principles and Applications by Ronal J. Tocci and Neal S. Widmer, PHI
6. Digital Fundamentals by Floyd, Pearson Education

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember the principles and functions of combinational and sequential digital circuits, data converters, and logic families.
- Understand the operation of adders, multiplexers, decoders, flip-flops, counters, registers, and converters in digital systems.
- Apply logic design techniques to realize combinational and sequential circuits for specific functional requirements.
- Analyze circuit performance considering speed, fan-in/fan-out, noise margin, and scalability across SSI, MSI, LSI, and VLSI technologies.
- Create integrated digital solutions by combining multiple circuit components for real-world computational and control applications.

CS-MJ-P-2	**Digital System Design Lab**	Semester	Credit	Hours	Evaluation
Major		II	2	30	60+10+5
Practical					

Course Objectives

To equip students with the skills to design, implement, and test combinational and sequential digital circuits using basic logic gates and standard ICs, enabling them to develop functional digital subsystems for computational and control applications.

Suggested Experiments

Combinational Circuits

- Implement Half Adder/Half Subtractor / Full Adder / Full Subtractor using Logic Gates. Realize a logic function using basic/universal gates in SOP and POS form. Study the functionalities of 7483 and design a BCD adder using 7483 or equivalent.
- Design a 4 bit 2's complement adder – subtractor unit using 7483 or equivalent and XOR gates.
- Design a circuit to convert BCD numbers to corresponding gray codes.
- Design a 4:1 MUX using NAND gates. Study of 74153 and 74151. Design Full Adder / Subtractor using MUX.
- Design a 2:4 decoder using NAND gates. Study of 74155 and 74138. Design Full Adder / Subtractor using decoders.
- Design a parity generator/checker using basic gates.
- Design magnitude comparator using basic/universal gates. Study of 7485.
- Design a seven-segment display unit.

Sequential Circuits

- Realize S-R, D, J-K and T flip-flop using basic gates. (Study the undefined state in S-R flip- flop).
- Study the functional characteristic of IC 74194 with emphasis on timing diagram.
- Design Asynchronous and Synchronous counters. (Mod-8, Mod-10 up counter).
- Study the functional characteristics of RAM IC chip. Study of open collector and tri- state output. Horizontal expansion of RAM chips by cascading. (Use 74189, 7489, or any available chip).

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design and implement combinational circuits such as adders, subtractors, multiplexers, decoders, comparators, code converters, parity generators, and display units using basic and universal logic gates as well as standard ICs.
- Apply logic design techniques to realize arithmetic and logic functions using MSI components like 7483, 74153, 74151, 74138, and 7485.
- Develop and analyze sequential circuits including flip-flops, shift registers, and counters, and interpret their timing diagrams for correct functionality.
- Interface and expand memory devices (RAM ICs) for higher capacity through cascading techniques, while understanding tri-state and open-collector outputs.
- Integrate combinational and sequential building blocks to create functional digital subsystems suitable for computational and control applications.

CS-MI-T-2	**Computer Fundamentals and Programming using C**	Semester	Credit	Hours	Evaluation
Minor		II	3	30	10+25
Theory					

Course Objectives

This course aims to provide students with a solid foundation in computer fundamentals, data representation, Boolean algebra, and problem-solving techniques, along with an introduction to the C programming language for developing efficient algorithms and implementing structured solutions to computational problems.

Syllabus

Unit I	**Introduction to Computer and Problem Solving** Information and Data. Hardware: CPU, Primary and Secondary storage, I/O devices Software: Systems and Application.	**3 Hrs.**
Unit II	**Introduction to Programming Languages** Machine Language, Assembly Language, High Level Language. Problem Solving: Flow Charts, Decision Tables and Pseudo codes.	**3 Hrs.**
Unit III	**Number Representation** Weighted Codes, Non-weighted codes, Positional, Binary, Octal, Hexadecimal, Binary Coded Decimal (BCD), Conversion of bases. Complement notions. Binary Arithmetic, Binary Codes: Gray, Alphanumeric, ASCII, EBCDIC; Parity Bits. Single Error-Detecting and Correcting Codes, Hamming Codes, Fixed and Floating Point Arithmetic.	**4 Hrs.**
Unit IV	**Boolean Algebra** Fundamentals of Boolean Algebra, Switches and Inverters, Functionally Complete Gates (AND, OR, NOT), NAND, NOR. Switching function and Boolean function. De Morgan's theorem, Minterm and Maxterm, Truth table and minimization of switching function up to four variables, Algebraic and K-map method of logic circuit synthesis: Two level and Multi level.	**4 Hrs.**
Unit V	**Introduction to C** C character set, Identifiers and keywords, Data types, Declarations, Expressions, statements and symbolic constants. Input-Output - getchar, putchar, scanf, printf, gets, puts, functions. Pre-processor commands - #include, #define, #ifdef	**6 Hrs.**
Unit VI	**Operators, expressions and storage Types** Arithmetic, unary, logical, bit-wise, assignment and conditional operators. Storage types - Automatic, external, register and static variables.	**3 Hrs.**
Unit VII	**Functions** Defining and accessing, passing arguments, Function prototypes, Recursion, Library functions, Static functions.	**3 Hrs.**
Unit VIII	**Arrays and Strings** Arrays- Defining and processing, Passing arrays to a function, Multi dimensional arrays. Strings- Defining and operations on strings.	**4 Hrs.**

Recommended Books

1. P. K. Sinha & Priti Sinha , —Computer Fundamentalsll, BPB Publications, 2007.
2. Dr. Anita Goel, Computer Fundamentals, Pearson Education, 2010.
3. Kernighan, Brian W., and Dennis M. Ritchie.The C programming language. 2006.
4. Schildt, Herbert, and C. Turbo. "C: the complete reference, Osborne." (2000).
5. Balagurusamy, E. programming in ANSI C. Tata McGraw-Hill Education, 2002.
6. Kanetkar, Yashavant P. Let us C. BPB publications, 2016.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember – Recall the basic concepts of computer hardware, software, number systems, Boolean algebra, and C language syntax.
- Understand – Explain problem-solving techniques using flowcharts, decision tables, pseudo codes, and describe data representation and logic simplification methods.
- Apply – Implement algorithms in C using variables, operators, functions, arrays, and strings to solve computational problems.
- Analyze – Differentiate between storage types, operators, and programming constructs; simplify logic expressions using Boolean algebra and Karnaugh maps.
- Evaluate – Assess the efficiency and correctness of algorithms and C programs through debugging and testing.
- Create – Design and develop structured C programs incorporating modular programming concepts, recursion, and data handling.

CS-MJ-P-2	**Programming using C Lab**	Semester	Credit	Hours	Evaluation
Minor		II	1	30	15
Practical					

Course Objectives

This course aims to build a strong foundation in computer fundamentals, problem-solving, and C programming, covering number systems, Boolean algebra, and core programming constructs such as functions, arrays and pointers enabling students to design and implement efficient solutions to real-world problems.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

Introduction to Computer and Problem Solving

(No direct coding experiments — focus on problem solving & logic design exercises in the lab before programming)

Draw **flowcharts** and write **pseudocode** for:

- Finding the area and circumference of a circle.
- Finding the largest of three numbers.
- Checking whether a number is prime.
- Demonstrate simple algorithms using **decision tables**.

Introduction to C programming, Number Representation & Boolean Algebra

1. Write a C program to convert a number from decimal to binary, octal, and hexadecimal.
2. Write a C program to convert binary to decimal.
3. Write a C program to perform binary addition, subtraction, multiplication, and division.
4. Write a C program to find the **1's and 2's complement** of a binary number.
5. Write a C program to find the **Gray code** of a given binary number and vice versa.
6. Write a C program to read the radius of a circle and find its area and circumference.
7. Write a C program to read three numbers and find the largest among them.
8. Write a C program to demonstrate library functions in *math.h*.
9. Write a C program to check whether a number is prime.
10. Write a C program to generate the first *n* prime numbers.
11. Write a C program to read a number, find the sum of its digits, reverse the number, and check whether it is a palindrome.
12. Write a C program to read numbers continuously until the user enters 999, and find the sum of only positive numbers.
13. Write a C program to read percentage of marks and display an appropriate message (else-if ladder).
14. Write a C program to find the roots of a quadratic equation (switch-case statement).

Operators, Expressions & Storage Types

- Modify earlier programs to use:
- Logical, bitwise, and conditional operators.
- Static and register variables to observe lifetime and scope.

Functions

1. Write a C program to check whether a number is prime by defining an *isprime*() function.
2. Write a C program to find factorial using recursion.
3. Write a C program to find GCD and LCM using functions.
4. Write a C program to read, display, and find the trace of a square matrix.
5. Write a C program to read, display, and add two $m \times n$ matrices using functions.
6. Write a C program to read, display, and multiply two matrices using functions.

Arrays and Strings

1. Write a C program to read marks scored by *n* students and find the average.
2. Write a C program to remove duplicate elements from a single-dimensional array.
3. Write a C program to demonstrate string functions in C.
4. Write a C program to read a string and count alphabets, digits, vowels, consonants, spaces, and special characters.

Pointers and Structures

1. Write a C program to demonstrate pointers in C.
2. Write a C program to reverse a string using pointers.
3. Write a C program to swap two numbers using pointers.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design algorithms using flowcharts and pseudocode for basic computational problems.
- Write and debug C programs using core concepts like functions, arrays, pointers, structures, and file.
- Apply programming to solve mathematical, logical, and data processing tasks.
- Demonstrate practical proficiency in C through hands-on lab exercises.

CS-MU-P-2	**Office Automation**	Semester	Credit	Hours	Evaluation
Multidisciplinary		II	3	30	10+35
Practical					

Course Objectives

To develop students' proficiency in using word processors, spreadsheets, and presentation tools for document creation, data analysis, visualization, and multimedia integration, enabling them to produce professional-quality reports, presentations, and analytical outputs.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

1. **Open an existing word document or create a new one**
   - Insert a table with a specific number of rows and columns.
   - Enter data into the table and format it (e.g., adjusting cell width, changing cell background color).
   - Add a new row and column to the existing table. Create a simple line graph based on the table data.
   - Edit the graph to add axis labels, a title, and customize the appearance. Add a header with your name and student ID.
   - Insert a footer with the document title and page count. Customize the header and footer layout and formatting.
   - Adjust the page numbering format (e.g., Roman numerals, starting from a specific page).
2. **Prepare a sample file in Spreadsheet or CSV format, including fields like "Name," "Address," and "Email."**
   - Create a template for a standard letter.
   - Perform a mail merge to personalize the letter with information from the file.
   - Review and edit merged documents to ensure correct personalization. Print the personalized letters or save them as separate documents.
3. Create a personal budget spreadsheet that tracks income, expenses, and savings over a specified period. Use formulas and functions to calculate totals, percentages, and remaining balances.
4. Choose a topic of your own interest and prepare a presentation according to the following instructions:

   Slide 1 – Title Slide

   Add title & subtitle, Apply background color, shadow, or 3D effects to both title & subtitle

   Slide 2 – Bulleted List

   Insert bulleted list slide, Add heading and points

   Slide 3 – Table or Chart

   Insert table/chart slide, Replace with your own data

   Slide 4 – Graphic or Word Art

   Use blank slide, Insert graphic/word art, text box, shapes with colors & effects

   Slide 5 – Any Layout

   Free choice of layout and content
5. Incorporate multimedia elements and graphics into a presentation.

   Tasks:
   - Insert a relevant photo/image onto a specific slide in the presentation. Embed a video clip from an external source (e.g., YouTube) into another slide. Add background audio or sound effect to the presentation.
   - Ensure proper alignment and resizing of multimedia elements.
   - Test the multimedia elements to ensure they play correctly during a slideshow.
6. A dataset containing sales data for a company be provided. Create a spreadsheet that calculates monthly sales totals, identifies top-selling products, and visualizes sales trends using line charts or bar graphs. Use conditional formatting to highlight exceptional sales performances.
7. Loan parameters, such as principal amount, interest rate, and loan term to be provided. Create a spreadsheet that calculates monthly loan payments, remaining balances, and interest paid over time using appropriate formulas. Create a chart to visualize the loan's repayment schedule.
8. Dataset to be provided which will allow various data analysis tasks using spreadsheets. Calculation of summary statistics, sorting and filtering data, creating pivot tables for deeper insights, and generation of charts or graphs to visualize patterns or trends within the data.
9. A dataset containing time-series data for multiple variables (e.g., monthly sales data for different products) to be provided and the following task to be performed; to create a combo chart with lines and columns to compare the trends of the variables and identify any relationships or patterns.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Create and format professional documents with tables, charts, headers, footers, and page numbering using word processing software.
- Perform mail merge operations by integrating data from spreadsheets or CSV files to generate personalized documents.
- Design and manage spreadsheets for budgeting, sales analysis, loan calculations, and general data analysis using formulas, functions, charts, and conditional formatting.
- Prepare and deliver multimedia-rich presentations incorporating tables, charts, graphics, audio, and video elements with effective formatting.
- Analyze and visualize datasets using pivot tables, combo charts, and various graphical methods to identify trends, patterns, and performance metrics.

CS-SEC-P-2	**Web Development and Applications Lab**	Semester	Credit	Hours	Evaluation

Skill Enhancement Practical		II	3	30	35+5+5

Course Objectives

This course is intended to give the students an introduction to the Hypertext Markup Language and its various components. Use of different HTML components like links, images, tables, headers, forms, CSS, etc., to design an ergonomic and efficient webpage will be taught in this course.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

1. Create HTML document with following formatting – Bold, Italics, Underline, Colors, Headings, Title, Font and Font Width, Background, Paragraph, Line Brakes, Horizontal Line, Blinking text as well as marquee text.
2. Create HTML document with Ordered and Unordered lists, Inserting Images, Internal and External linking.
3. Create HTML document with Table:

				Some Image Here

4. Create Form with Input Type, Select and Text Area in HTML.
5. Create an HTML containing Roll No., student's name and Grades in a tabular form.
6. Create an HTML document (having two frames) which will appear as follows:

About	This frame would show the contents according to the link clicked by the user on the left frame.
Department 1	
Department 2	
Department 3	

7. Create an HTML document containing horizontal frames as follows:

Department Names (could be along with Logos)
Contents according to the Link clicked

8. Create a website of 6 – 7 pages with different effects as mentioned in above problems.

List of Practical Assignments to be done using Javascript

1. Print a table of numbers from 5 to 15 and their squares and cubes using alert.
2. Print the largest of three number
3. Find the factorial of a number n.
4. Enter a list of positive numbers terminated by Zero. Find the sum and average of these numbers.
5. A person deposits Rs 1000 in a fixed account yielding 5% interest. Compute the amount in the account at the end of each year for n years.
6. Read n numbers. Count the number of negative numbers, positive numbers and zeros in the list.

Suggested Readings

Text Books:

1. Learning Web Design: A Beginner`s Guide To HTML, CSS, JavaScript, And Web Graphics
2. D.R. Brooks, An Introduction to HTML and Javascript for Scientists and Engineers, Springer W. Willard,2009

Reference Books/Links:

1. HTML A Beginner's Guide, Tata McGraw-Hill Education, 2009
2. J. A. Ramalho, Learn Advanced HTML 4.0 with DHTML, BPB Publications, 2007
3. https://developer.mozilla.org/en-US/docs/Web/CSS4.
4. https://developer.mozilla.org/en-US/docs/Web/HTML
5. https://developer.mozilla.org/en-US/docs/Web/JavaScript

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design and format structured HTML pages using text formatting, colors, headings, images, lists, tables, forms, and frames.
- Develop multi-page websites with internal/external links, multimedia integration, and layout control using frames and styling.
- Implement interactive features in web pages using JavaScript for calculations, condition checks, loops, and data processing.
- Apply JavaScript logic to solve problems such as factorial computation, number classification, interest calculation, and statistical summaries.
- Integrate HTML and JavaScript to build functional, user-friendly web interfaces with both static content and dynamic behavior.

Detail Syllabus of Semester-III

CS-MJ-T-3	**Computer Organization & Architecture**	Semester	Credit	Hours	Evaluation
Major		III	6	60	15+60
Theory					

Course Objectives

To provide students with a strong foundation in digital logic, data representation, computer organization, CPU architecture, memory systems, and I/O mechanisms, enabling them to understand, design, and analyze the functional components of a computer system.

Syllabus

Unit I	**Introduction** Logic gates, Boolean algebra, combinational circuits, circuit simplification, flip-flops and sequential circuits, decoders, multiplexers, registers, counters and memory units.	**4 Hrs.**
Unit II	**Data Representation and Basic Computer Arithmetic** Number systems, complements, fixed and floating point representation, character representation, addition, subtraction, magnitude comparison, multiplication and division algorithms for integers.	**10 Hrs.**
Unit III	**Basic Computer Organization and Design** Computer registers, bus system, instruction set, timing and control, instruction cycle, memory reference, input-output and interrupt, Interconnection Structures, Bus Interconnection design of basic computer.	**10 Hrs.**
Unit IV	**Central Processing Unit** Register organization, arithmetic and logical micro-operations, stack organization, micro programmed control. Instruction formats, addressing modes, instruction codes, machine language, assembly language, input output programming, RISC, CISC architectures, pipelining and parallel architecture.	**18 Hrs.**
Unit V	**Memory Organization** Cache memory, Associative memory, mapping.	**10 Hrs.**
Unit VI	**Input-Output Organization** Input / Output: External Devices, I/O Modules, Programmed I/O, Interrupt-Driven I/O, Direct Memory Access, I/O Channels.	**8 Hrs.**

Recommended Books

1. M. Mano, Computer System Architecture, Pearson Education 1992
2. A. J. Dos Reis, Assembly Language and Computer Architecture using C++ and JAVA, Course Technology, 2004
3. W. Stallings, Computer Organization and Architecture Designing for Performance, 8th Edition, Prentice Hall of India,2009
4. M.M. Mano , Digital Design, Pearson Education Asia,2013
5. Carl Hamacher, Computer Organization, Fifth edition, McGrawHill, 2012.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember fundamental concepts of logic gates, Boolean algebra, number systems, computer organization, and memory structures.
- Understand the design and operation of combinational/sequential circuits, CPU architectures, and I/O systems.
- Apply algorithms for arithmetic operations, instruction execution, and control in basic computer design.
- Analyze CPU organization, addressing modes, and interconnection structures to evaluate performance trade-offs between RISC, CISC, pipelining, and parallel processing.
- Evaluate memory mapping strategies, cache performance, and I/O mechanisms for optimized system performance.
- Create simplified digital and computer organization designs integrating logic circuits, CPU control, memory, and I/O subsystems.

CS-MI-T-3	**Database Management Systems**	Semester	Credit	Hours	Evaluation
Minor		III	3	30	10+25
Theory					

Course Objectives

To enable students to understand database concepts, design principles, relational models, SQL programming, normalization techniques, transaction processing, and indexing, equipping them to design, implement, and manage efficient and secure database systems.

Syllabus

Unit I	**Introduction** Concept & Overview of DBMS, Data Models, Database Languages, Database Administrator, Database Users, Three Schema architecture of DBMS. Characteristics of database approach, data models, database system architecture and data independence.	**3 Hrs.**
Unit II	**Entity Relationship(ER) Modeling** Basic concepts, Design Issues, Mapping Constraints, Keys, Entity-Relationship Diagram, Weak Entity Sets, Extended E-R features.	**4 Hrs.**
Unit III	**Relation data model** Structure of relational Databases, Relational Algebra, Relational Calculus, Extended Relational Algebra Operations, Views, Modifications of the Database.	**4 Hrs.**
Unit IV	**SQL and Integrity Constraints** Concept of DDL, DML, DCL. Basic Structure, Set operations, Aggregate Functions, Null Values, Domain Constraints, Referential Integrity Constraints, assertions, views, Nested Subqueries, Database security application development using SQL, Stored procedures and triggers.	**5 Hrs.**
Unit V	**Relational Database Design** Functional Dependency, Different anomalies in designing a Database., Normalization using functional dependencies, Decomposition, Boyce-Codd Normal Form, 3NF, Normalization using multi-valued dependencies, 4NF, 5NF.	**4 Hrs.**
Unit VI	**Internals of RDBMS** Physical data structures, Query optimization: join algorithm, statistics and cost bas optimization. Transaction processing.	**3 Hrs.**
Unit VII	**Transaction Processing** ACID properties, Concurrency control and Recovery Management : transaction model properties, state serializability, lock base protocols, two phase locking.	**4 Hrs.**
Unit VIII	**In File Structure and Indexing** Operations on files, File of Unordered and ordered records, overview of File organizations, Indexing structures for files (Primary index, secondary index, clustering index), Multilevel indexing using B and B+ trees.	**3 Hrs.**

Recommended Books

1. Henry F. Korth and Silberschatz Abraham, —Database System Conceptsǁ, Mc.Graw Hill.
2. Elmasri Ramez and Novathe Shamkant, —Fundamentals of Database Systemsǁ, Benjamin Cummings Publishing. Company.
3. Ramakrishnan: Database Management System, McGraw-Hill
4. Date C. J., —Introduction to Database Managementǁ, Vol. I, II, III, Addison Wesley.
5. Ullman JD., —Principles of Database Systemsǁ, Galgottia Publication.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember fundamental concepts of DBMS, data models, three-schema architecture, and database system components.
- Understand ER modeling, relational algebra/calculus, SQL commands, and integrity constraints for database creation and manipulation.
- Apply SQL queries, stored procedures, triggers, and normalization techniques to design and manage relational databases.
- Analyze database design issues, functional dependencies, anomalies, and performance aspects such as query optimization and indexing.
- Evaluate concurrency control protocols, recovery mechanisms, and data security measures to ensure reliability and integrity.
- Create optimized, normalized, and secure relational databases using advanced SQL features, indexing methods, and transaction management strategies.

CS-MI-P-3	**Database Management Systems Lab**	Semester	Credit	Hours	Evaluation
Minor		III	1	20	15
Practical					

Course Objectives

To develop the ability to design, create, and manage relational databases, retrieve and manipulate data using SQL, enforce integrity constraints, and apply normalization techniques, enabling students to build efficient and reliable database systems.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

1. **Creating Database**:
   Creating a Database Creating a Table
   Specifying Relational Data Types Specifying Constraints Creating Indexes
2. **Table and Record Handling**:
   INSERT statement
   Using SELECT and INSERT together DELETE, UPDATE, TRUNCATE statements DROP, ALTER statements
3. **Retrieving Data from a Database:**
   Use the SELECT statement to query data.
   Apply the WHERE clause with logical operators, IN, BETWEEN, and LIKE for filtering.
   Organize results using ORDER BY, GROUP BY, and HAVING.
   Perform calculations with aggregate functions.
4. Specify conditions using Boolean and comparison operators (AND, OR, NOT, =, <>, >, <, >=, <=).
5. Use arithmetic operators and aggregate functions (COUNT, SUM, AVG, MIN, MAX).
6. Perform multiple table queries (joins on different or same tables).
7. Write nested SELECT statements.
8. Manipulate sets using (ANY, IN, CONTAINS, ALL, NOT IN, NOT CONTAINS, EXISTS, NOT EXISTS, UNION, INTERSECT, MINUS/EXCEPT, etc.).
9. Create a database and tables for a simple student management system.
10. Insert, update, delete, and retrieve data using SQL commands.
11. Use SELECT with WHERE, ORDER BY, GROUP BY, and HAVING clauses.
12. Create tables with PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, and CHECK constraints.
13. Demonstrate ALTER TABLE to modify schema and constraints.
14. Retrieve specific records using SELECT with DISTINCT, LIKE, and BETWEEN operators.
15. Use aggregate functions (COUNT, SUM, AVG, MIN, MAX) with GROUP BY.
16. Implement UNION, INTERSECT, and EXCEPT operations.
17. Perform join operations: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, and SELF JOIN.
18. Write queries using IN, EXISTS, ANY, ALL, and correlated subqueries.
19. Retrieve data from multiple tables using nested queries.
20. Create and manipulate VIEWS to simplify complex queries.
21. Implement transactions using COMMIT, ROLLBACK, and SAVEPOINT.
22. Convert unnormalized tables into 1NF, 2NF, and 3NF.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember SQL syntax for creating databases, defining tables, specifying data types, and applying constraints.
- Understand how to retrieve, filter, group, and sort data using various SQL clauses, logical operators, and aggregate functions.
- Apply SQL commands for inserting, updating, deleting, and modifying records and table structures.
- Analyze queries involving multiple tables, joins, subqueries, set operations, and views for solving complex data retrieval problems.
- Evaluate database designs by enforcing primary/foreign keys, constraints, and normalization up to 3NF to ensure data integrity.
- Create functional database solutions with transaction control, indexing, and optimized queries for real-world applications like a student management system.

CS-MU-T-3	**AI for Everyone**	Semester	Credit	Hours	Evaluation
Multidisciplinary		III	3	30	10+35
Theory					

Course Objectives

To introduce students to the fundamental concepts, subfields, and applications of Artificial Intelligence, while fostering an understanding of its ethical, social, and future implications, enabling them to critically evaluate and responsibly engage with AI technologies.

Syllabus

Unit I	**Introduction to Artificial Intelligence** Definition and scope of AI; historical overview and key milestones; differentiating AI from human intelligence.	**3 Hrs.**
Unit II	**AI Subfields and Technologies** Introduction and basic concepts of machine learning, including supervised, unsupervised, and reinforcement learning, deep learning and neural networks (without technical details); basic concepts of natural language processing (NLP) and computer vision.	**8 Hrs.**
Unit III	**Applications of AI** AI in healthcare (diagnosis, treatment, medical imaging); AI in finance (fraud detection, algorithmic trading, risk assessment); AI in transportation (autonomous vehicles, traffic optimization); AI in education (personalized learning, intelligent tutoring systems).	**8 Hrs.**
Unit IV	**Ethical and Social Implications of AI** Bias and fairness in AI systems; privacy and data protection concerns; impact of AI on employment and the workforce; AI and social inequality.	**7 Hrs.**
Unit V	**Emerging Issues and Future Trends** Ethical guidelines and responsible AI practices; AI and innovation; emerging trends and future directions in AI; AI and creativity (generative models, artistic applications).	**4 Hrs.**

Recommended Books

1. S. Goswami, A. K. Das, A. Chakrabarti, “AI for Everyone: A Beginner's Handbook for Artificial Intelligence (AI)”, Pearson, 2024.
2. P. Verdegem, “AI for Everyone?: Critical Perspectives”, University of Westminster Press, 2021.
3. Shawn Schuster, “AI For All: How Everyday People Can Benefit from Artificial Intelligence”, UMLAUT Publishing, 2021.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Explain the scope, history, and key concepts of Artificial Intelligence.
- Describe basic principles of machine learning, deep learning, NLP, and computer vision.
- Identify and discuss AI applications in various domains.
- Examine ethical, social, and privacy issues related to AI.
- Assess emerging trends and responsible AI practices for future developments.

CS-MI-P-3	**Data Analysis through Python/R Lab**	Semester	Credit	Hours	Evaluation
Skill Enhancement		III	3	30	35+5+5
Practical					

Course Objectives

To equip students with skills in preprocessing, analyzing, and visualizing data using Python or R, applying statistical and machine learning techniques to real-world datasets. Students will develop practical abilities in extracting, interpreting, and deriving meaningful insights, with emphasis on effectively communicating results through compelling visualizations and well-structured reports for data-driven problem-solving in academic, research, and industry contexts.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

1. Write a Python/R script to create, manipulate, and perform basic operations on lists (Python) or vectors (R). Perform operations such as slicing, indexing, and appending elements.
2. Write a Python/R program to load and manipulate data using dictionaries (Python) or data frames (R).
3. Write a Python/R script to load a CSV file into a Pandas DataFrame (Python) or a data frame (R). Perform basic operations like viewing the first few rows, summary statistics, and exporting the modified dataset to a new CSV file. Import data from an Excel file and perform similar operations.
4.
5. Write a Python/R script to identify and handle missing data (drop or impute missing values) in a dataset. Detect and treat outliers using statistical techniques. Perform data normalization or standardization on numerical columns.
6. Perform univariate and multivariate analysis using Python (Pandas, Matplotlib, Seaborn) or R (ggplot2, dplyr).
7. Generate summary statistics (mean, median, mode, standard deviation, etc.) for numerical columns. Create visualizations such as histograms, boxplots, scatter plots, and pair plots to explore relationships between variables.
8. (a) Write a Python/R script to apply log transformations, binning, or scaling to numerical data.
9. Create new features using existing ones (e.g., adding a column for a calculated field).
10. Perform one-hot encoding for categorical variables and label encoding for target variables.
11. Write a Python/R script to compute and visualize the correlation matrix for a dataset. Perform hypothesis testing using t-tests or ANOVA to compare means between groups. Apply chi-square tests for independence on categorical data.
12. Create line plots, bar plots, pie charts, and heatmaps using Python (Matplotlib/Seaborn) or R (ggplot2).
13. Create advanced visualizations like violin plots, KDE plots, and facet grids to represent multi- dimensional data.
14. Customize plots by adding titles, labels, legends, and changing color schemes.
15. Implement a simple linear regression model in Python (Scikit-learn) or R to predict a target variable. Evaluate the model using metrics like R-squared, mean squared error, and visualize the regression line. Extend to multiple linear regression and evaluate its performance.
16. Implement a logistic regression model in Python/R to classify binary data. Evaluate model performance using confusion matrix, accuracy, precision, recall, and F1-score. Explore other classification algorithms like Decision Trees and K-Nearest Neighbors (KNN) for comparison.
17. Implement the K-means clustering algorithm in Python/R to group data based on similarity. Visualize clusters and calculate the silhouette score to evaluate cluster quality. Apply hierarchical clustering and compare the results with K-means.
18. Load and visualize time series data using Python (Pandas) or R. Perform decomposition of time series into trend, seasonality, and residuals. Implement ARIMA or exponential smoothing to forecast future data points.
19. Write a Python/R script to perform PCA on a dataset and reduce its dimensions. Visualize the explained variance and transformed data. Use the reduced dataset to perform further analysis or machine learning.
20. Implement cross-validation techniques to evaluate model performance. Use techniques like GridSearchCV (Python) or tune. grid (R) to find the best hyperparameters for models. Compare multiple models and choose the best one based on performance metrics.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember SQL syntax for creating databases, defining tables, specifying data types, and applying constraints.
- Understand how to retrieve, filter, group, and sort data using various SQL clauses, logical operators, and aggregate functions.
- Apply SQL commands for inserting, updating, deleting, and modifying records and table structures.
- Analyze queries involving multiple tables, joins, subqueries, set operations, and views for solving complex data retrieval problems.
- Evaluate database designs by enforcing primary/foreign keys, constraints, and normalization up to 3NF to ensure data integrity.
- Create functional database solutions with transaction control, indexing, and optimized queries for real-world applications like a student management system.

Detail Syllabus of Semester-IV

CS-MJ-T-4	**Data Structures**	Semester	Credit	Hours	Evaluation
Major		IV	4	40	15+60
Theory					

Course Objectives

To provide students with a comprehensive understanding of linear and non-linear data structures, their representations, and associated algorithms for searching, sorting, hashing, and file handling. The course aims to develop problem-solving skills through the design, analysis, and implementation of efficient data structures, enabling students to choose appropriate techniques for real-world applications.

Syllabus

Unit I	**Introduction** Basic definitions; classifications; ADT; Linear Data Structures - Sequential representations; Non- linear data structures – representations	**2 Hrs.**
Unit II	**Arrays** Single and Multi-dimensional Arrays; Sparse Matrices (Array and Linked Representation); Row- major and column-major order; different applications	**2 Hrs.**
Unit III	**Stacks** Implementing single / multiple stack/s in an Array; Prefix, Infix and Postfix expressions, Utility and conversion of these expressions from one to another; Evaluation of postfix expression using stack; Applications of stack; Limitations of Array representation of stack.	**3 Hrs.**
Unit IV	**Linked Lists** Singly, Doubly and Circular Lists (Array and Linked representation); Normal and Circular representation of Stack in Lists; Self Organizing Lists; Skip Lists.	**4 Hrs.**
Unit V	**Queues** Array and Linked representation of Queue, De-queue, Priority Queues	**2 Hrs.**
Unit VI	**Recursion** Developing Recursive Definition of Simple Problems and their implementation; Advantages and Limitations of Recursion; Understanding what goes behind Recursion (Internal Stack Implementation); Tail Recursion; When not to use recursion; Removal of recursion.	**4 Hrs.**
Unit VII	**Trees and Graphs** Introduction to Tree as a data structure; Binary Trees (Insertion, Deletion , Recursive and Iterative Traversals on Binary Search Trees); Different properties of Binary trees; Threaded Binary Trees (Insertion, Deletion, Traversals); Height-Balanced Trees (Various operations on AVL Trees); B- tree, B+ tree; Graphs - Representations, Breadth-first and Depth-first Search.	**6 Hrs.**
Unit VIII	**Searching and Sorting** Linear Search, Binary Search, Comparison of Linear and Binary Search; Bubble Sort, Selection Sort, Insertion Sort, Heap Sort, Merge Sort, Radix Sort, Bucket Sort, Shell Sort; Comparison of Sorting Techniques.	**10 Hrs.**
Unit IX	**Hashing** Introduction to Hashing, Deleting from Hash Table, Efficiency of Rehash Methods, Hash Table Reordering, Resolving collision by Open Addressing, Coalesced Hashing, Separate Chaining, Dynamic and Extendible Hashing, Choosing a Hash Function, Perfect Hashing Function	**4 Hrs.**
Unit X	**File Structures** Sequential and Direct Access. Relative Files, Indexed Files - B+ tree as index. Multi-indexed Files, Inverted Files, Hashed Files.	**3 Hrs.**

Recommended Books

1. Adam Drozdek, "Data Structures and algorithm in C++", Third Edition, Cengage Learning, 2012.
2. Sartaj Sahni, Data Structures, "Algorithms and applications in C++", Second Edition, Universities Press, 2011.
3. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using C and C++:, Second edition, PHI, 2009.
4. Robert L. Kruse, "Data Structures and Program Design in C++", Pearson,1999.
5. Robert Lafore, "Data Structures and Algorithms in Java, 2/E", Pearson/ Macmillan Computer Pub,2003
6. John Hubbard, "Data Structures with JAVA", McGraw Hill Education (India) Private Limited; 2 edition, 2009

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember fundamental concepts, classifications, and representations of linear and non-linear data structures, including arrays, stacks, queues, linked lists, trees, graphs, and files.
- Understand operations, properties, and applications of various data structures, recursion techniques, and algorithmic strategies.
- Apply appropriate data structures and algorithms for implementing and solving computational problems such as expression evaluation, searching, sorting, and traversal.
- Analyze the efficiency, advantages, and limitations of data structures and algorithms, including time and space complexity considerations.
- Evaluate alternative data structure designs, hashing methods, and file organization techniques for optimal performance in specific contexts.
- Create efficient and maintainable solutions by integrating suitable data structures and algorithms for given problem scenarios.

CS-MJ-P-4	**Data Structures Lab**	Semester	Credit	Hours	Evaluation
Major		IV	2	30	60+10+5
Practical					

Course Objectives

To develop students' ability to implement, manipulate, and apply various linear and non-linear data structures using templates in C++, and to enhance problem-solving skills through recursion, iteration, and efficient algorithm design. The course aims to prepare students to choose and apply suitable data structures for solving real-world computational problems.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2. WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.
4. Implement Doubly Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
5. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation. Use Templates.
8. Perform Queues operations using Circular Array implementation. Use Templates.
9. Create and perform different operations on Double-ended Queues using Linked List implementation.
10. WAP to scan a polynomial using linked list and add two polynomial.
11. WAP to calculate factorial and to compute the factors of a given no.
    (i)using recursion, (ii) using iteration
12. WAP to display Fibonacci series
    (i)using recursion, (ii) using iteration
13. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion
14. WAP to create a Binary Search Tree and include following operations in tree:
    - Insertion (Recursive and Iterative Implementation)
    - Deletion by copying
    - Deletion by Merging
    - Search a no. in BST
    - Display its preorder, postorder and inorder traversals Recursively
    - Display its preorder, postorder and inorder traversals Iteratively
    - Display its level-by-level traversals
    - Count the non-leaf nodes and leaf nodes
    - Display height of tree
    - Create a mirror image of tree
    - Check whether two BSTs are equal or not
15. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
16. WAP to reverse the order of the elements in the stack using additional stack.
17. WAP to reverse the order of the elements in the stack using additional Queue.
18. WAP to implement Diagonal Matrix using one-dimensional array.
19. WAP to implement Lower Triangular Matrix using one-dimensional array.
20. WAP to implement Upper Triangular Matrix using one-dimensional array.
21. WAP to implement Symmetric Matrix using one-dimensional array.
22. WAP to create a Threaded Binary Tree as per inorder traversal and implement operations like finding the successor / predecessor of an element, insert an element, inorder traversal.
23. WAP to implement various operations on AVL Tree.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember syntax, operations, and properties of common data structures such as arrays, linked lists, stacks, queues, trees, and matrices.
- Understand the use of templates for creating generic implementations of data structures and algorithms.
- Apply linear and binary search, various sorting algorithms, and linked list operations for dynamic data handling.
- Analyze recursive and iterative approaches for solving problems such as factorial, Fibonacci series, and GCD computation.
- Evaluate different tree structures (BST, Threaded Binary Tree, AVL) for performance in insertion, deletion, searching, and traversal operations.
- Create efficient C++ programs implementing advanced data structures (circular linked list, double-ended queue, sparse matrix, triangular matrices, symmetric matrix) to solve domain-specific problems.

CS-MJ-T-5	**Discrete Mathematics**	Semester	Credit	Hours	Evaluation
Major		IV	6	60	15+60
Theory					

Course Objectives

To provide students with a solid foundation in discrete mathematical structures, including set theory, combinatorics, recurrence relations, graph theory, and propositional logic, enabling them to model, analyze, and solve computational problems using rigorous mathematical reasoning.

Syllabus

Unit I	**Introduction** Sets - finite and Infinite sets, uncountably Infinite Sets; functions, relations, Properties of Binary Relations, Closure, Partial Ordering Relations; counting - Pigeonhole Principle, Permutation and Combination; Mathematical Induction, Principle of Inclusion and Exclusion.	**8 Hrs.**
Unit II	**Growth of Functions** Asymptotic Notations, Summation formulas and properties, Bounding Summations, approximation by Integrals.	**12 Hrs.**
Unit III	**Recurrences** Recurrence Relations, generating functions, Linear Recurrence Relations with constant coefficients and their solution, Substitution Method, Recurrence Trees, Master Theorem.	**15 Hrs.**
Unit IV	**Graph Theory** Basic Terminology, Models and Types, multigraphs and weighted graphs, Graph Representation, Graph Isomorphism, Connectivity, Euler and Hamiltonian Paths and Circuits, Planar Graphs, Graph Coloring, Trees, Basic Terminology and properties of Trees, Introduction to Spanning Trees.	**15 Hrs.**
Unit V	**Propositional Logic** Logical Connectives, Well-formed Formulas, Tautologies, Equivalences, Inference Theory.	**10 Hrs.**

Recommended Books

1. C.L. Liu , D.P. Mahopatra, Elements of Discrete mathematics, 2nd Edition , Tata McGraw Hill, 1985,
2. Kenneth Rosen, Discrete Mathematics and Its Applications, Sixth Edition ,McGraw Hill 2006
3. T.H. Coremen, C.E. Leiserson, R. L. Rivest, Introduction to algorithms, 3rd edition Prentice Hall on India, 2009
4. M. O. Albertson and J. P. Hutchinson, Discrete Mathematics with Algorithms , John wiley Publication, 1988
5. J. L. Hein, Discrete Structures, Logic, and Computability, 3rd Edition, Jones and Bartlett Publishers, 2009
6. D.J. Hunter, Essentials of Discrete Mathematics, Jones and Bartlett Publishers, 2008

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember fundamental concepts of sets, functions, relations, counting principles, asymptotic notations, recurrence relations, graphs, trees, and propositional logic.
- Understand properties of relations, counting techniques, growth of functions, recurrence solving methods, and graph theoretical models.
- Apply combinatorial techniques, asymptotic analysis, recurrence solutions, and logical reasoning to solve discrete mathematics problems.
- Analyze graph connectivity, planarity, coloring, spanning trees, and logical equivalences to determine problem feasibility and complexity.
- Evaluate the suitability of mathematical models and methods for representing and solving real-world computational problems.
- Create mathematical formulations and proofs for problems in computer science using discrete structures and logic.

CS-MI-T-4	**Database Management Systems**	Semester	Credit	Hours	Evaluation
Minor		III	3	30	10+25
Theory					

Course Objectives

To enable students to understand database concepts, design principles, relational models, SQL programming, normalization techniques, transaction processing, and indexing, equipping them to design, implement, and manage efficient and secure database systems.

Syllabus

Unit I	**Introduction** Concept & Overview of DBMS, Data Models, Database Languages, Database Administrator, Database Users, Three Schema architecture of DBMS. Characteristics of database approach, data models, database system architecture and data independence.	**3 Hrs.**
Unit II	**Entity Relationship(ER) Modeling** Basic concepts, Design Issues, Mapping Constraints, Keys, Entity-Relationship Diagram, Weak Entity Sets, Extended E-R features.	**5 Hrs.**
Unit III	**Relation data model** Structure of relational Databases, Relational Algebra, Relational Calculus, Extended Relational Algebra Operations, Views, Modifications of the Database.	**5 Hrs.**
Unit IV	**SQL and Integrity Constraints** Concept of DDL, DML, DCL. Basic Structure, Set operations, Aggregate Functions, Null Values, Domain Constraints, Referential Integrity Constraints, assertions, views, Nested Subqueries, Database security application development using SQL, Stored procedures and triggers.	**5 Hrs.**
Unit V	**Relational Database Design** Functional Dependency, Different anomalies in designing a Database., Normalization using functional dependencies, Decomposition, Boyce-Codd Normal Form, 3NF, Normalization using multi-valued dependencies, 4NF, 5NF.	**7 Hrs.**
Unit VI	**Internals of RDBMS** Physical data structures, Query optimization: join algorithm, statistics and cost bas optimization. Transaction processing.	**3 Hrs.**
Unit VII	**Transaction Processing** ACID properties, Concurrency control and Recovery Management : transaction model properties, state serializability, lock base protocols, two phase locking.	**5 Hrs.**
Unit VIII	**File Structure and Indexing** Operations on files, File of Unordered and ordered records, overview of File organizations, Indexing structures for files (Primary index, secondary index, clustering index), Multilevel indexing using B and B+ trees.	**5 Hrs.**

Recommended Books

1. Henry F. Korth and Silberschatz Abraham, —Database System Conceptsǁ, Mc.Graw Hill.
2. Elmasri Ramez and Novathe Shamkant, —Fundamentals of Database Systemsǁ, Benjamin Cummings Publishing. Company.
3. Ramakrishnan: Database Management System, McGraw-Hill
4. Date C. J., —Introduction to Database Managementǁ, Vol. I, II, III, Addison Wesley.
5. Ullman JD., —Principles of Database Systemsǁ, Galgottia Publication.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember fundamental concepts of DBMS, data models, three-schema architecture, and database system components.
- Understand ER modeling, relational algebra/calculus, SQL commands, and integrity constraints for database creation and manipulation.
- Apply SQL queries, stored procedures, triggers, and normalization techniques to design and manage relational databases.
- Analyze database design issues, functional dependencies, anomalies, and performance aspects such as query optimization and indexing.
- Evaluate concurrency control protocols, recovery mechanisms, and data security measures to ensure reliability and integrity.
- Create optimized, normalized, and secure relational databases using advanced SQL features, indexing methods, and transaction management strategies.

CS-MI-P-4	**Database Management Systems Lab**	Semester	Credit	Hours	Evaluation
Minor		III	1	20	15
Practical					

Course Objectives

To develop the ability to design, create, and manage relational databases, retrieve and manipulate data using SQL, enforce integrity constraints, and apply normalization techniques, enabling students to build efficient and reliable database systems.

List of Suggested Practical / Laboratory Experiments to be conducted but not limited to the following:

1. **Creating Database**:
   Creating a Database Creating a Table
   Specifying Relational Data Types Specifying Constraints Creating Indexes
2. **Table and Record Handling**:
   INSERT statement
   Using SELECT and INSERT together DELETE, UPDATE, TRUNCATE statements DROP, ALTER statements
3. **Retrieving Data from a Database:**
   Use the SELECT statement to query data.
   Apply the WHERE clause with logical operators, IN, BETWEEN, and LIKE for filtering.
   Organize results using ORDER BY, GROUP BY, and HAVING.
   Perform calculations with aggregate functions.
4. Specify conditions using Boolean and comparison operators (AND, OR, NOT, =, <>, >, <, >=, <=).
5. Use arithmetic operators and aggregate functions (COUNT, SUM, AVG, MIN, MAX).
6. Perform multiple table queries (joins on different or same tables).
7. Write nested SELECT statements.
8. Manipulate sets using (ANY, IN, CONTAINS, ALL, NOT IN, NOT CONTAINS, EXISTS, NOT EXISTS, UNION, INTERSECT, MINUS/EXCEPT, etc.).
9. Create a database and tables for a simple student management system.
10. Insert, update, delete, and retrieve data using SQL commands.
11. Use SELECT with WHERE, ORDER BY, GROUP BY, and HAVING clauses.
12. Create tables with PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, and CHECK constraints.
13. Demonstrate ALTER TABLE to modify schema and constraints.
14. Retrieve specific records using SELECT with DISTINCT, LIKE, and BETWEEN operators.
15. Use aggregate functions (COUNT, SUM, AVG, MIN, MAX) with GROUP BY.
16. Implement UNION, INTERSECT, and EXCEPT operations.
17. Perform join operations: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, and SELF JOIN.
18. Write queries using IN, EXISTS, ANY, ALL, and correlated subqueries.
19. Retrieve data from multiple tables using nested queries.
20. Create and manipulate VIEWS to simplify complex queries.
21. Implement transactions using COMMIT, ROLLBACK, and SAVEPOINT.
22. Convert unnormalized tables into 1NF, 2NF, and 3NF.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Remember SQL syntax for creating databases, defining tables, specifying data types, and applying constraints.
- Understand how to retrieve, filter, group, and sort data using various SQL clauses, logical operators, and aggregate functions.
- Apply SQL commands for inserting, updating, deleting, and modifying records and table structures.
- Analyze queries involving multiple tables, joins, subqueries, set operations, and views for solving complex data retrieval problems.
- Evaluate database designs by enforcing primary/foreign keys, constraints, and normalization up to 3NF to ensure data integrity.
- Create functional database solutions with transaction control, indexing, and optimized queries for real-world applications like a student management system.

Detail Syllabus of Semester-V

CS-MJ-T-6	**Database Management Systems**	Semester	Credit	Hours	Evaluation
Major		V	4	40	15+60
Theory					

Course Objectives

The objective of this Database Management Systems (DBMS) course is to provide students with a comprehensive understanding of core database concepts, models, and architectures, enabling them to systematically analyze, design, and implement relational databases using ER modelling, relational algebra, and SQL. The course aims to equip students with the knowledge of database normalization, internal data storage structures, indexing, and transaction management, while emphasizing database security, integrity constraints, and practical skills required for developing robust and efficient data-driven applications in real-world scenarios.

Syllabus

Unit I	**Introduction** Concept & Overview of DBMS, Data Models, Database Languages, Database Administrator, Database Users, Three Schema architecture of DBMS. Characteristics of database approach, data models, database system architecture and data independence	**2 Hrs.**
Unit II	**Entity Relationship(ER) Modelling** Basic concepts, Design Issues, Mapping Constraints, Keys, Entity-Relationship Diagram, Weak Entity Sets, Extended E-R features.	**5 Hrs.**
Unit III	**Relation data model** Structure of relational Databases, Relational Algebra, Relational Calculus, Extended Relational Algebra Operations, Views, Modifications Of the Database.	**4 Hrs.**
Unit IV	**SQL and Integrity Constraints** Concept of DDL, DML, DCL. Basic Structure, Set operations, Aggregate Functions, Null Values, Domain Constraints, Referential Integrity Constraints, assertions, views, Nested Subqueries, Database security application development using SQL, Stored procedures and triggers.	**6 Hrs.**
Unit V	**Relational Database Design** Functional Dependency, Different anomalies in designing a Database., Normalization using functional dependencies, Decomposition, Boyce-Codd Normal Form, 3NF, Normalization using multi-valued dependencies, 4NF, 5NF.	**8 Hrs.**
Unit VI	**Internals of RDBMS** Physical data structures, Query optimization: join algorithm, statistics and cost bas optimization. Transaction processing.	**5 Hrs.**
Unit VII	**Transaction Processing** ACID properties, Concurrency control and Recovery Management : transaction model properties, state serializability, lock base protocols, two phase locking.	**5 Hrs.**
Unit VIII	**File Structure and Indexing** Operations on files, File of Unordered and ordered records, overview of File organizations, Indexing structures for files (Primary index, secondary index, clustering index).	**5 Hrs.**

Recommended Books

1. Henry F. Korth and Silberschatz Abraham, —Database System Concepts‖, Mc.Graw Hill.
2. Elmasri Ramez and Novathe Shamkant, —Fundamentals of Database Systems‖, Benjamin Cummings Publishing
3. Ramakrishnan: Database Management System , McGraw-Hill
4. Date C. J., —Introduction to Database Management‖, Vol. I, II, III, Addison Wesley.
5. Ullman JD., —Principles of Database Systems‖, Galgottia Publication.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Explain DBMS concepts, architectures, data models, and user roles.
- Design conceptual database schemas using ER modelling and relational mapping.
- Construct and optimize SQL queries for definition, manipulation, and retrieval.
- Apply normalization (up to BCNF/4NF) to reduce redundancy and anomalies.
- Analyze internal storage, file organization, and indexing for efficient data access.
- Understand transaction management, concurrency control, and recovery mechanisms.
- Implement security, integrity constraints, and develop data-driven applications.

CS-MJ-P-6	**Database Management Systems Lab**	Semester	Credit	Hours	Evaluation
Major		V	2	30	60+10+5
Practical					

Course Objectives

The objective of this SQL module is to equip students with a comprehensive, hands-on understanding of Structured Query Language for relational database management—covering the creation and management of databases and tables, data manipulation and retrieval using various commands and clauses, implementation of constraints, security, user controls, and advanced programming with PL/SQL—enabling them to design, query, and administer databases effectively in real-world applications.

Suggested Programs

1. **Entity–Relationship (ER) Modeling**
   - Draw ER diagrams for a real-world application (e.g., Library Management, College Database, Online Shopping System).
   - Identify entities, relationships, mapping cardinalities/constraints, and convert ER diagram to relational schema.
2. **Relational Schema & Table Creation**
   - Create relational database tables from the derived schema.
   - Define appropriate data types, primary and foreign keys, and constraints (NOT NULL, UNIQUE, CHECK, DEFAULT, etc.).
3. **SQL Data Manipulation**
   - Write SQL queries to INSERT, UPDATE, DELETE, and SELECT data from single and multiple tables.
   - Use aggregate functions (SUM, AVG, COUNT, MIN, MAX) and GROUP BY/HAVING clauses.
   - Retrieve data using advanced queries: nested subqueries, set operations (UNION, INTERSECT), and JOIN operations (INNER JOIN, LEFT/RIGHT JOIN, FULL JOIN).
4. **Views, Indexing, and Security**
   - Create, update, and drop VIEWS.
   - Implement INDEXES to optimize queries.
   - Create users; implement GRANT and REVOKE privileges for access control.
5. **Integrity Constraints and Triggers**
   - Enforce domain, entity, and referential integrity with appropriate constraints such as CHECK, PRIMARY KEY, FOREIGN KEY, UNIQUE, and assertions (where supported).
   - Create and test database TRIGGERS to automatically perform actions on INSERT, UPDATE, or DELETE events; for example, maintaining audit logs or enforcing complex integrity rules.
   - Implement and experiment with CURSORS to perform row-by-row processing of query results within procedural code (e.g., PL/SQL or T-SQL). Tasks may include:
     - Declaring and opening cursors.
     - Fetching rows one at a time and processing them.
     - Using cursors in triggers or stored procedures to handle complex business logic that requires iterative operations on dataset rows.
6. **Stored Procedures & Functions**
   - Write and execute SQL stored procedures and functions for complex processing (e.g., calculate employee salary, grade calculation, inventory updates).

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Create databases, tables, define data types, and enforce constraints.
- Manage records with INSERT, SELECT, UPDATE, DELETE, and ALTER.
- Retrieve and filter data using SELECT with advanced clauses.
- Use aggregate functions, joins, and subqueries for complex queries.
- Implement database security by managing users and privileges.
- Simplify queries with views and column aliases for readability.
- Develop and manage PL/SQL stored procedures and cursors.

CS-MJ-T-7	**Design and Analysis of Algorithms**	Semester	Credit	Hours	Evaluation
Major		VI	6	60	15+60
Theory					

Course Objectives

The objective of this course is to develop students' understanding of the design, analysis, and implementation of algorithms by emphasizing space and time complexity, correctness, and efficiency; to introduce key algorithmic design paradigms including greedy, divide and conquer, dynamic programming, and backtracking; and to cultivate the ability to rigorously analyze and compare algorithms for a variety of classic problems in searching, sorting, graph theory, and computational complexity, equipping students to identify, formulate, and solve problems using appropriate algorithmic techniques in computing.

Syllabus

Unit I	**Introduction & Analysis Technique** Definition, Characteristics, Correctness of Algorithm, Recursive and Non-recursive algorithms. Space and Time Complexity, Efficiency of an algorithm, Growth of Functions, Polynomial and Exponential Complexity, Asymptotic Notations: Big O Notation, Big Ω Notations, Big Θ Notations, Properties: Best case/worst case/average case analysis of well-known algorithms.	**14 Hrs.**
Unit II	**Searching and Sorting Techniques** Elementary Searching (Linear and Binary) and sorting techniques–Bubble Sort, Insertion Sort, Selection Sort, Advanced Sorting techniques- Merge Sort, Heap Sort, Quick Sort, Sorting in Linear Time- Bucket Sort; Complexity Analysis.	**10 Hrs.**
Unit III	**Algorithm Design Techniques** Concepts and simple case studies of Greedy algorithms. Divide and conquer: Basic concepts, Case study of selected searching and sorting problems as divide and conquer techniques, Dynamic programming: General issues in Dynamic Programming, Case study of Binomial Coefficient computation. Backtracking Algorithms, Case study of N-Queen problem.	**16 Hrs.**
Unit IV	**Graph Representation and Algorithm** Graph traversal algorithms: Breadth First Search, Depth First Search and its Applications, Minimal spanning trees: Prim's Algorithm, Kruskal's Algorithm, Shortest path algorithms: Floyd's Algorithm, Floyd-Warshall Algorithm, Dijkstra's Algorithm, Bellman-Ford algorithm, Graph Coloring Algorithms. Application of Tree, Decision Trees, Red-Black Trees.	**12 Hrs.**
Unit V	**Classification of Problems:** P, NP, Satisfiability, Cook's Theorem (Statement Only).	**8 Hrs.**

Recommended Books

1. T.H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein Introduction to Algorithms, PHI, 3rd Edition 2009.
2. Sara Basse & A.V. Gelder Computer Algorithm– Introduction to Design and Analysis, Publisher– Pearson 3rd Edition 1999
3. Fundamentals of Computer Algorithms, E.Horowitz and Sahani, Galgoti
4. Algorithm Design, Jon Kleiberg and Eva Tardos, Pearson Education.
5. Data Structures and Algorithms- K.Mehlhorn , EATCS, Vol. I & Vol. 2

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Analyze time and space complexity using asymptotic notations.
- Prove recursive and non-recursive algorithms' correctness with induction.
- Compare and apply sorting and searching algorithms' complexities.
- Use algorithm paradigms (greedy, divide and conquer, dynamic programming).
- Implement graph algorithms (BFS, DFS, Dijkstra, MST) for applications.
- Classify problems as P, NP, and NP-complete, discuss theory.
- Address algorithm limitations using approximation and decision-tree methods.

CS-MJ-T-8	**Microprocessor and Its Applications**	Semester	Credit	Hours	Evaluation
Major		V	4	40	15+60
Theory					

Course Objectives

The objective of this course is to provide students with a solid foundation in microprocessor architectures, focusing on the 8085 and 8086, and to develop practical skills in assembly language programming, hardware interfacing, and peripheral device control. The course aims to equip learners with the knowledge to analyze and design microprocessor-based systems by mastering processor architecture, instruction sets, programming techniques, and interfacing methods, preparing them for advanced study or careers in embedded systems and hardware-oriented computing.

Syllabus

Unit I	**Introduction to Microprocessors & 8085 Assembly Language Programming** Microprocessors – Instruction set and computer languages – 8085 programming model – Instruction classification – Instruction – Data format and storage.	**2 Hrs.**
Unit II	**8085 Microprocessor architecture** Microprocessor Architecture and its operations – Memory – I/O Devices, 8085 MPU – 8085 based microcomputer – memory interfacing – 8155 memory segment Interfacing – Interfacing I/O devices: Basics – Interfacing input and output devices – memory mapped I/O.	**8 Hrs.**
Unit III	**Programming 8085** Instruction Set of 8085 – Data Transfer – arithmetic – Logic – Branch – Writing ALP and Debugging programs – Looping – Counting and Indexing – 16-bit Arithmetic instructions – Logic operations – Counters and Time Delay.	**9 Hrs.**
Unit IV	**Interfacing I/O Devices** Stack and subroutines – Restart – Conditional call and Return instruction – Advanced subroutine concepts – Code conversion – BCD Arithmetic and 16-bit operations – BCD- Binary conversion – Binary to BCD conversion – BCD to seven segment LED code conversion – Binary to ASCII and ASCII to binary conversion – BCD addition and subtraction.	**6 Hrs.**
Unit V	**Interfacing Peripheral (I/O) and Applications** Interrupts: 8085 Interrupt – RST instructions – Software and Hardware interrupt – multiple Interrupts and Priorities – 8085 Vectored Interrupts – Restart as Software Instructions – 8155 – Multipurpose programmable Device – 8279 Programmable Keyboard/Display Interface – 8255 Programmable peripheral Interface, Interrupts and DMA. Peripherals: 8279, 8255, 8251, 8253, 8237, 8259, A/D and D/A converters and interfacing of the same.	**9 Hrs.**
Unit VI	**Introduction to 8086** 16 bit processors: 8086 and architecture, segmented memory has cycles, read/write cycle in min/max mode. Reset operation, wait state, Halt state, Hold state, Lock operation, interrupt processing. Addressing modes and their features. Software instruction set (including specific instructions like string instructions, repeat, segment override, lock prefizers and their use).	**6 Hrs.**

Recommended Books

1. Microprocessor architecture, programming and applications with 8085/8085A, Wiley eastern Ltd, 1989 by Ramesh S. Gaonkar.
2. Intel Corp: The 8085 / 8085A. Microprocessor Book – Intel marketing communication, Wiley inter science publications, 1980.
3. An introduction to micro computers Vol. 2 – some real Microprocessor – Galgotia Book Source, New Delhi by Adam Osborne and J. Kane
4. Advanced Microprocessors by Ray and Bhurchandi - TMH
5. Intel Corp. Micro Controller Handbook – Intel Publications, 1994.
6. Assembly Language Programming the IBM PC by Alan R. Miller, Subex Inc, 1987
7. The Intel Microprocessors: 8086/8088, 80186, 80286, 80386 & 80486, Bary B. Brey, Prentice Hall, India 1996

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Describe architectural features, components, instruction sets, and programming models of 8085/8086.
- Write and debug 8085 assembly programs using data transfer and arithmetic instructions.
- Analyze memory and I/O interfacing with external devices and programmable peripherals.
- Implement stack operations, subroutines, and code conversion techniques for program flow.
- Handle interrupts and DMA, including external, internal, vectored, and prioritized interrupts.
- Develop microprocessor-based applications by interfacing peripherals and troubleshooting hardware.
- Understand 8086 architecture, addressing modes, instruction set, and advanced software features.

CS-MJ-P-8	**Microprocessor and Its Applications Lab**	Semester	Credit	Hours	Evaluation
Major		V	2	30	60+10+5
Practical					

Course Objectives

The objective of this course is to provide students with practical experience in microprocessor programming and interfacing. Students will learn to perform basic arithmetic and data transfer operations, logical operations, and branching, along with implementing code conversion and interfacing techniques. The course will also cover stack and subroutine operations, generating delays, and interacting with peripheral devices for real-world embedded applications.

Suggested 8085 Assembly Language Programs

1. **Basic Data Transfer and Arithmetic**
   - Program to add/subtract/multiply/divide two 8-bit or 16-bit numbers.
   - Move a block of data from one memory location to another.
   - Generate the 1's and 2's complement of an 8-bit number.
2. **Logical Operations**
   - Count the number of 1's (set bits) in an 8-bit number.
   - Find the largest/smallest number in an array.
3. **Branching and Looping**
   - Generate Fibonacci series up to n terms.
   - Sum of a series of numbers stored in memory.
   - Sort an array in ascending/descending order (e.g., Bubble Sort).
4. **Code Conversion**
   - Convert binary to BCD and vice versa.
   - Convert binary to ASCII and ASCII to binary.
   - Convert BCD to seven-segment code for display.
5. **Counters, Delays, and Timers**
   - Generate a time delay using a loop.
   - Program for up/down counter.
6. **Stack and Subroutine Operations**
   - Use of PUSH, POP, CALL, and RET instructions with single/multiple stack frames.
7. **Interfacing Simulation**
   - Interact with 8255 PPI to display an incrementing count on LEDs.
   - Keyboard input using switches and display on LEDs/Seven Segment.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Perform basic arithmetic operations and data transfer between memory locations.
- Execute logical operations such as counting set bits and identifying the largest or smallest number in an array.
- Generate a Fibonacci series, sum numbers, and implement sorting algorithms like Bubble Sort.
- Convert data between binary, BCD, ASCII, and seven-segment display codes.
- Implement time delays, up/down counters, and timers in microprocessor programming.
- Use stack operations (PUSH, POP) and subroutines (CALL, RET) with multiple stack frames.
- Interact with the 8255 PPI for LED display and process input from switches.

CS-MI-P-5	**Programming In Python**	Semester	Credit	Hours	Evaluation
Minor		V	4	40	35+10+5
Practical					

Course Objectives

The objective of this course is to introduce students to fundamental programming concepts and problem-solving using Python. Students will gain hands-on experience writing and debugging programs involving input/output, control structures, functions, data structures, object-oriented principles, file handling, and basic data visualization.

Suggested Programs

1. Introduction to Computers and Programming
   - Install Python and run a simple program that prints a greeting message.
   - Write a program to demonstrate: print() function with formatting and use of comments.
   - Create a program that takes input from the user, performs arithmetic calculations, and displays results with formatting.
   - Write a program that takes name and age as input and displays a personalized message.
   - Create a temperature converter (Celsius ↔ Fahrenheit) using input/output operations.
2. Decision and Repetition Structures
   - Write a program using if and if-else statements to check whether a number is even or odd, and positive or negative, String comparison to check for login credentials (username/password match).
   - Nested decision making: Determine student grade based on marks.
   - Use of for and while loop to display numbers from 1 to N and calculate factorial, multiplication table.
   - Program for input validation (e.g., age must be between 1–120).
   - Create a program with nested loops to print patterns (e.g., triangle of stars '*').
3. Functions, Lists, Tuples, Plotting
   - Define and call simple functions (void and value-returning).
   - Program to demonstrate: Local vs Global variables; Function arguments (positional, keyword, default).
   - Generate and return a random number using random module.
   - Recursive function to compute factorial or Fibonacci numbers.
   - Create and manipulate a list: add, delete, find max/min, sort.
   - Use list slicing and looping to reverse a list and copy it.
   - Work with 2D list (matrix): Sum of rows, columns, or diagonal elements.
   - Tuple operations: define, access, and unpack values.
   - Plot list data using matplotlib (e.g., plot marks of students vs names).
4. Strings, Dictionaries, Files, OOP
   - String operations: length, slicing, search (in), replace, count vowels.
   - Use of dictionaries to store and retrieve student records.
   - Set operations: union, intersection, difference, membership testing.
   - File operations: Write a program to read from and write to a file; Count number of words or lines in a text file. Handle exceptions during file reading and input processing.
   - Basic Object-Oriented Programming: Create a class Student with attributes and methods.
   - Create instances, access attributes, and invoke methods.
   - Program demonstrating inheritance and method overriding.
   - Polymorphism example using method overloading or overriding.

Recommended Books

1. Python Programming: An Introduction to Computer Science by John Zelle (3rd Edition, 2017, Franklin, Beedle & Associates) – A beginner-friendly book introducing Python and computer science fundamentals.
2. Python for Data Analysis by Wes McKinney (2nd Edition, 2017, O'Reilly Media) – Focuses on Python for data analysis using the pandas library.
3. Python Data Science Handbook by Jake VanderPlas (1st Edition, 2016, O'Reilly Media) – A comprehensive guide to essential Python tools for data analysis, machine learning, and visualization.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Write Python programs using print(), input(), and basic arithmetic operations.
- Implement decision structures (if, if-else) and loops (while, for).
- Define and use functions, including recursion, with local and global variables.
- Manipulate lists, tuples, and 2D lists, including sorting, slicing, and searching.
- Perform string operations and utilize dictionaries and sets for data storage.
- Read from and write to files, with proper exception handling.
- Apply object-oriented programming concepts such as classes, inheritance, and polymorphism.

Detail Syllabus of Semester-VI

CS-MJ-T-9	**Data Communication and Computer Networks**	Semester	Credit	Hours	Evaluation
Major		VI	6	60	15+60
Theory					

Course Objectives

This syllabus aims to provide students with a solid understanding of computer networks, including data communication, networking protocols, and network architecture. Topics include data representation, network types (LAN, MAN, WAN), OSI/TCP-IP models, signal modulation, multiplexing, error handling, network switching, multiple access protocols, routing techniques, and transport-layer protocols. The course emphasizes practical applications in real-world scenarios, preparing students for network design, management, and troubleshooting across various network layers.

Syllabus

Unit I	**Introduction to Computer Networks** Introduction; Data communications: components, data representation (ASCII,ISO etc.),direction of data flow(simplex, half duplex, full duplex); Networks: distributed processing, network criteria, physical structure (type of connection, topology), categories of network (LAN, MAN,WAN);Internet: brief history, internet today; Protocols and standards; Reference models: OSI reference model, TCP/IP reference model, their comparative study.	**8 Hrs.**
Unit II	**Data Communication Fundamentals and Techniques** Analog and digital signal; data-rate limits; digital to digital line encoding schemes; pulse code modulation; parallel and serial transmission; digital to analog modulation; multiplexing techniques-FDM, TDM; transmission media.	**10 Hrs.**
Unit III	**Networks Switching Techniques and Access mechanisms** Message switching; Circuit switching; Packet switching- connectionless datagram switching, connection-oriented virtual circuit switching; dial-up modems; digital subscriber line; cable TV for data transfer.	**8 Hrs.**
Unit IV	**Data Link Layer Functions and Protocol** Error detection and error correction techniques; data-link control- framing and flow control; error recovery protocols- stop and wait ARQ, go-back-n ARQ; Point to Point Protocol on Internet; HDLC.	**8 Hrs.**
Unit V	**Multiple Access Protocol and Networks** CSMA/CD protocols; Ethernet LANS; connecting LAN and back-bone networks- repeaters, hubs, switches, bridges, router and gateways FDDI, token bus, token ring; Reservation, polling, concentration; Multiple access protocols: Pure ALOHA, Slotted ALOHA.	**10 Hrs.**
Unit VI	**Networks Layer Functions and Protocols** Internetworking & devices: Repeaters, Hubs, Bridges, Switches, Router, Gateway; Addressing : Internet address, classful address, subnetting; Routing: techniques, static vs. dynamic routing, routing table for classful address; Routing algorithms: shortest path algorithm, flooding, distance vector routing, link state routing; Protocols: ARP, RARP, IP, ICMP, IPV6; Unicast and multicast routing protocols.	**8 Hrs.**
Unit VII	**Transport Layer Functions and Protocols** Process to process delivery; UDP; TCP; Congestion control algorithm: Leaky bucket algorithm, Token bucket algorithm, choke packets; Quality of service: techniques to improve QoS.	**6 Hrs.**
Unit VIII	**Overview of Application layer protocol** Overview of DNS; SMTP, SNMP, FTP, HTTP & WWW.	**2 Hrs.**

Recommended Books

1. "Computer Networks" by Andrew S. Tanenbaum and David J. Wetherall, published by Pearson, 6th Edition (2017).
2. "Data and Computer Communications" by William Stallings, published by Pearson, 11th Edition (2019).
3. "Computer Networking: A Top-Down Approach" by James F. Kurose and Keith W. Ross, published by Pearson, 8th Edition (2021).
4. "Networking All-in-One For Dummies" by Doug Lowe, published by Wiley, 8th Edition (2021).
5. "The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference" by Charles M. Kozierok, published by No Starch Press, 1st Edition (2005).
6. "Computer Networks and Internets" by Douglas E. Comer, published by Pearson, 7th Edition (2021).

Outcome

- **Understand Networking Concepts**: Learn data communication components, network types, and the OSI/TCP-IP models.
- **Data Communication Techniques**: Apply modulation, encoding, and multiplexing for efficient transmission.
- **Network Switching**: Compare message, circuit, and packet switching techniques.
- **Data Link Layer**: Implement error handling, flow control, and ARQ protocols.
- **Routing and Addressing**: Master IP addressing, subnetting, and routing algorithms.
- **Transport & Application Protocols**: Apply TCP, UDP, and protocols like DNS, SMTP, and HTTP.

CS-MJ-T-10	**Operating Systems**	Semester	Credit	Hours	Evaluation
Major		VI	6	60	15+60
Theory					

Course Objectives

This course provides a comprehensive overview of operating systems (OS), focusing on core concepts such as process management, memory management, file and I/O management, and protection and security. It covers process life cycles, scheduling algorithms, process synchronization, deadlock handling, and interprocess communication. Students will learn about CPU scheduling techniques like FCFS, SJF, RR, and priority scheduling, and understand the fundamentals of memory management, including paging, segmentation, and virtual memory. The course also delves into file systems, disk management, and I/O techniques like DMA, polling, and interrupts. Additionally, it explores system security, including authentication, encryption, and threat monitoring, equipping students with a well-rounded understanding of OS structure and functionality.

Syllabus

Unit I	**Overview and Process management Introduction** Operating System Structures – Operating systems services – System calls. Process Management: Concept of processes; Process states; PCB; process scheduling; operations on processes; co- operating processes; interprocess communication, Threads Overview, benefits of threads, user and kernel threads	**8 Hrs.**
Unit II	**Scheduling algorithms and Process Synchronization** CPU scheduling: scheduling criteria, preemptive & non-preemptive scheduling, scheduling algorithms (FCFS, SJF, RR, priority), algorithm evaluation, multi-processor scheduling. Process Synchronization: Background, critical section problem, critical region, synchronization hardware, classical problems of synchronization, semaphores; monitor. Deadlocks: system model, deadlock characterization, methods for handling deadlocks, deadlock prevention, deadlock avoidance, deadlock detection, recovery from deadlock.	**16 Hrs.**
Unit III	**Memory Management** Physical and virtual address space; swapping; memory allocation strategies - fixed and variable partitions; paging; segmentation; segmentation with paging; virtual memory; demand paging; page replacement algorithms and their relative advantages and disadvantages; thrashing.	**16 Hrs.**
Unit IV	**File and I/O Management** File concept, access methods, directory structure, file system structure, allocation methods (contiguous, linked, indexed), free-space management (bit vector, linked list, grouping), directory implementation (linear list, hash table), efficiency & performance. I/O hardware, polling, interrupts, DMA, application I/O interface (block and character devices, network devices, clocks and timers, blocking and nonblocking I/O), kernel I/O subsystem (scheduling, buffering, caching, spooling and device reservation, error handling), performance. Disk management: Disk structure, disk scheduling (FCFS, SSTF, SCAN,C-SCAN) , disk reliability, disk formatting, boot block, bad blocks.	**14 Hrs.**
Unit V	**Protection and Security** Goals of protection, domain of protection, security problem, authentication, one time password, program threats, system threats, threat monitoring, encryption.	**6 Hrs.**

Recommended Books

1. A. Silberschatz, P. B. Galvin, G. Gagne, Operating System Concepts, Addison Wesley.
2. W. Stalling, Operating Systems: Internals and Design Principles, PHI.
3. A. S. Tanenbaum, Modern operating Systems, PHI.
4. M. Milenkovic, Operating Systems- Concepts and design, Tata McGraw Hill 1992.
5. G. Nutt, Operating Systems: A Modern Perspective, 2nd Edition Pearson Education 1997.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Interpret Operating System Structure, Operations, Services and Process.
- Elaborate Multithreaded Programming, Process Scheduling and Synchronization.
- Evaluate different memory management schemes.
- Design and implement File system functionalities.
- Experiment with various disk management schemes.

CS-MJ-T-11	**Object Oriented Programming using JAVA**	Semester	Credit	Hours	Evaluation
Major		VI	4	40	15+60
Theory					

Course Objectives

The objective of this course is to provide students with a comprehensive understanding of object-oriented programming principles and practices using Java. The course aims to develop the ability to model and solve real-world problems by designing robust, modular, and reusable software systems. Through both theory and hands-on programming, students will gain proficiency in fundamental OOP concepts such as classes, objects, inheritance, polymorphism, exception handling, and file I/O, as well as exposure to advanced features like generics and multithreading, thus preparing them for higher studies or professional roles in contemporary software development.

Syllabus

Unit I	**Introduction to Object-Oriented Programming** Principles and features of OOP: **Abstraction**, **Encapsulation**, **Inheritance**, **Polymorphism**. Benefits, limitations, and applications of OOP. Procedural vs Object-Oriented Programming. Overview of the Java programming environment and IDEs (Integrated Development Environments) such as IntelliJ IDEA, Eclipse, or NetBeans.	**4 Hrs.**
Unit II	**Programming Basics in Java** Java program structure, syntax, and style. Keywords, identifiers, variables, constants, and data types. Input/Output operations using Scanner for console and file handling. Operators, expressions, and type casting in Java. Basic control structures in Java (if-else, switch, loops: for, while, do-while).	**4 Hrs.**
Unit III	**Classes and Objects in Java** Defining classes and objects in Java. Member data (instance variables) and methods (functions). Access specifiers: private, public, protected. Constructors and Destructors (Finalizers in Java: finalize() method). The this keyword in Java. Static members (static variables and methods). Method Overloading in Java (method signature, return type, and parameters).	**10 Hrs.**
Unit IV	**Inheritance and Polymorphism in Java** Types of inheritance: single inheritance, multilevel inheritance, and hierarchical inheritance. Method Overloading and Method Overriding in Java. Use of super keyword for calling parent class methods and constructors. Abstract classes and interfaces in Java. Polymorphism: Dynamic method dispatch, method overriding, and runtime polymorphism. Use of final keyword (for variables, methods, and classes).	**10 Hrs.**
Unit V	**Advanced Object-Oriented Features in Java** Exception Handling: try-catch blocks, exception propagation, and custom exceptions. File I/O: Streams, reading from and writing to files using FileReader, BufferedReader, FileWriter, and BufferedWriter. Generics in Java: Generic classes and methods. Threads and Multithreading: Thread creation using Runnable interface and Thread class. Packages in Java: Understanding and creating packages. Basic understanding of Java Collections Framework: List, Set, Map interfaces.	**12 Hrs.**

Recommended Books

1. **Balagurusamy**, "Object Oriented Programming with C++" & "Programming with JAVA".
2. **Herbert Schildt**, "C++: The Complete Reference" and "Java: The Complete Reference".
3. **Deitel & Deitel**, "C++ How to Program" / "Java How to Program".
4. **Stanley B. Lippman et al.**, "C++ Primer".
5. **Yashavant Kanetkar**, "Let Us C++".

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Solid foundational concepts of OOP—abstraction, encapsulation, inheritance, and polymorphism.
- Design and develop JAVA programs using classes, objects, and key OOP features.
- Implement programs utilizing inheritance, virtual functions, interfaces/abstract classes, and overloading.
- Apply file I/O, exception handling, and modular programming concepts in JAVA.
- Develop GUI-based and event-driven applications.
- Solve real-world problems using reusable, efficient, and robust object-oriented solutions.

CS-MJ-T-11	**Object Oriented Programming using C++**	Semester	Credit	Hours	Evaluation
Major		VI	4	40	15+60
Theory					

Course Objectives

The objective of this course is to provide students with a comprehensive understanding of object-oriented programming principles and practices using C++. The course aims to develop the ability to model and solve real-world problems by designing robust, modular, and reusable software systems. Through both theory and hands-on programming, students will gain proficiency in fundamental OOP concepts such as classes, objects, inheritance, polymorphism, exception handling, and file I/O, as well as exposure to advanced features like templates, memory management, and multithreading, thus preparing them for higher studies or professional roles in contemporary software development.

Syllabus

Unit I	**Introduction to Object-Oriented Programming in C++** Principles and features of OOP: Abstraction, Encapsulation, Inheritance, Polymorphism. Benefits, limitations, and applications of OOP. Procedural vs Object-Oriented Programming. Overview of the C++ programming environment and IDEs (Integrated Development Environments) such as Visual Studio, Code::Blocks, or CLion.	**4 Hrs.**
Unit II	**Programming Basics in C++** C++ program structure, syntax, and style. Keywords, identifiers, variables, constants, and data types. Input/Output operations using cin, cout, and file handling (ifstream, ofstream). Operators, expressions, type casting, and basic control structures (if-else, switch, loops: for, while, do-while).	**4 Hrs.**
Unit III	**Classes and Objects in C++** Defining classes and objects in C++. Member data (instance variables) and methods (functions). Access specifiers: private, public, protected. Constructors and Destructors in C++. The this pointer in C++. Static members (static variables and methods). Method Overloading in C++.	**10 Hrs.**
Unit IV	**Inheritance and Polymorphism in C++** Types of inheritance: single inheritance, multilevel inheritance, hierarchical inheritance, and multiple inheritance. Function/method overloading and method overriding in C++. Use of explicit pointer (this pointer) and dynamic binding. Virtual functions for runtime polymorphism. Dynamic Binding and Runtime Polymorphism in C++.	**10 Hrs.**
Unit V	**Advanced Object-Oriented Features in C++** Exception Handling: try-catch blocks, exception propagation, and custom exceptions. File I/O: Streams, reading from and writing to files using ifstream, ofstream, and fstream. Templates in C++: Template classes and functions for generic programming. Operator Overloading in C++. Introduction to Threads and Multithreading in C++ using std::thread. Namespaces in C++ for organizing code and avoiding name conflicts.	**12 Hrs.**

Recommended Books

1. Balagurusamy, "Object Oriented Programming with C++".
2. Herbert Schildt, "C++: The Complete Reference".
3. Deitel & Deitel, "C++ How to Program".
4. Stanley B. Lippman et al., "C++ Primer".
5. Yashavant Kanetkar, "Let Us C++".

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Solid foundational concepts of OOP—abstraction, encapsulation, inheritance, and polymorphism in C++.
- Design and develop C++ programs using classes, objects, and key OOP features.
- Implement programs utilizing inheritance, virtual functions, abstract classes, and overloading in C++.
- Apply file I/O, exception handling, and modular programming concepts in C++.
- Solve real-world problems using reusable, efficient, and robust object-oriented solutions in C++.

CS-MI-P-11	**Object Oriented Programming using JAVA Lab**	Semester	Credit	Hours	Evaluation
Major		VI	2	30	60+10+5
Practical					

Course Objectives

To provide hands-on experience in designing, implementing, and testing programs using core object-oriented principles with JAVA. To enable students to develop robust, modular, and reusable applications by applying concepts such as classes, inheritance, polymorphism, exception handling, and file I/O in practical scenarios.

Suggested Programs

1. Introduction to Object-Oriented Programming

- Write a program to print "Hello, World!" and demonstrate the structure of a Java program.
- Implement a simple procedural program and an equivalent OOP (class-based) version to compare approaches (such as calculating the area of geometric shapes).
- Write a program that lists and explains use of OOP concepts (Abstraction, Encapsulation, etc.) through comments and class diagrams.

2. Programming Basics in JAVA

- Program using variable declarations, input and output functions, arithmetic operations, and formatted output.
- Write a menu-driven program using control structures (if, switch, loops) for a calculator or student grade assessment.
- Demonstrate file I/O by reading data from a text file and displaying or processing its content.
- Program to illustrate type casting between data types and handling invalid conversions.

3. Classes and Objects

- Define a class with private data members and public methods; instantiate objects and access members.
- Implement constructors (default, parameterized) and finalize methods (to handle resource cleanup).
- Write a program using the 'this' keyword for resolving variable scope or chaining constructors.
- Demonstrate static data members and static member methods of a class.
- Implement method overloading for functions with the same name but different parameter lists.

4. Inheritance and Polymorphism

- Implement single, multilevel, and hierarchical inheritance. (Demonstrate multiple inheritance using interfaces.)
- Write a program to override methods/functions in derived classes (showing method overriding and use of the super keyword).
- Use dynamic method dispatch to show polymorphism in action.
- Create abstract classes and interfaces. Instantiate derived classes and invoke overridden methods.
- Demonstrate dynamic binding and runtime polymorphism by storing objects in a base class reference and calling overridden methods.

5. **Advanced Object-Oriented Features**

- Implement exception handling with try-catch blocks for common runtime errors (divide by zero, invalid input, array bounds).
- Create a custom exception (user-defined) and use it in an application.
- Write programs for file reading and writing: read from a file, write to a file, and copy file content.
- Implement a generic method/class for sorting or finding max/min in an array of any type.
- Write a simple multithreaded program (e.g., concurrent counter, printing numbers/strings in different threads).
- Use packages to organize code and prevent naming conflicts.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Apply core object-oriented concepts—classes, inheritance, and polymorphism—in JAVA.
- Develop and test modular applications with exception handling and file operations.
- Translate problem statements into reusable, well-structured code using modern programming practices.
- Organize code effectively using packages and programming tools.

CS-MI-P-11	**Object Oriented Programming using C++ Lab**	Semester	Credit	Hours	Evaluation
Major		VI	2	30	60+10+5
Practical					

Course Objectives

To provide hands-on experience in designing, implementing, and testing programs using core object-oriented principles with C++. To enable students to develop robust, modular, and reusable applications by applying concepts such as classes, inheritance, polymorphism, exception handling, and file I/O in practical scenarios.

Suggested Programs

1. **Introduction to Object-Oriented Programming**
   - Write a program to print "Hello, World!" and demonstrate the structure of a C++ program.
   - Implement a simple procedural program and an equivalent OOP (class-based) version to compare approaches (such as calculating the area of geometric shapes).
   - Write a program that lists and explains use of OOP concepts (Abstraction, Encapsulation, etc.) through comments and class diagrams.
2. **Programming Basics in C++**
   - Program using variable declarations, input and output functions, arithmetic operations, and formatted output.
   - Write a menu-driven program using control structures (if, switch, loops) for a calculator or student grade assessment.
   - Demonstrate file I/O by reading data from a text file and displaying or processing its content.
   - Program to illustrate type casting between data types and handling invalid conversions.
3. **Classes and Objects**
   - Define a class with private data members and public methods; instantiate objects and access members.
   - Implement constructors (default, parameterized, copy) and destructors.
   - Write a program using the 'this' pointer to resolve variable scope or chaining constructors.
   - Demonstrate static data members and static member methods of a class.
   - Implement method overloading for functions with the same name but different parameter lists.
4. **Inheritance and Polymorphism**
   - Implement single, multilevel, and hierarchical inheritance. (Also demonstrate multiple inheritance using multiple base classes in C++.)
   - Write a program to override methods/functions in derived classes (showing method overriding and use of explicit pointers).
   - Use virtual functions to show polymorphism in action.
   - Demonstrate dynamic binding and runtime polymorphism by storing objects in a base class pointer/reference and calling overridden methods.
5. **Advanced Object-Oriented Features**
   - Implement exception handling with try-catch blocks for common runtime errors (divide by zero, invalid input, array bounds).
   - Create a custom exception (user-defined) and use it in an application.
   - Write programs for file reading and writing: read from a file, write to a file, and copy file content.
   - Demonstrate operator overloading in C++ (such as overloading '+' for a complex number class).
   - Write a simple multithreaded program (e.g., concurrent counter, printing numbers/strings in different threads).
   - Use namespaces to organize code and prevent naming conflicts.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Apply core object-oriented concepts—classes, inheritance, and polymorphism—in **C++** programs.
- Develop and test modular applications with exception handling and file operations in C++.
- Translate problem statements into reusable, well-structured code using modern C++ programming practices.
- Organize code effectively using namespaces and C++ programming tools.

Detail Syllabus of Semester-VII

CS-MJ-T-12	**Advanced Mathematics**	Semester	Credit	Hours	Evaluation
Major		VII	6	60	15+60
Theory					

Course Objectives

This course covers optimization techniques like linear programming (simplex method, integer programming, transportation, and assignment problems), combinatorics and probability (permutations, combinations, Bayes' Theorem, random variables), and group theory (groups, subgroups, isomorphisms, rings, fields, and applications). It also includes matrix theory, determinants, systems of linear equations, eigenvalues, eigenvectors, and LU decomposition.

Syllabus

Unit I	**Optimization Techniques** Linear Programming – Mathematical Model, Graphical Solution, Simplex Method. Integer Programming, Transportation and Assignment Problems.	**15 Hrs.**
Unit II	**Combinatorics and Probability** Permutations and Combinations, Mathematical Induction, Probability, Bayes' Theorem. Random Variables and their distributions: Discrete and Continuous.	**15 Hrs.**
Unit III	**Group Theory** Groups, Subgroups, Semi Groups, Product and Quotients of Algebraic Structures, Isomorphism, Homomorphism, Automorphism, Rings, Integral Domains, Fields, Applications of Group Theory.	**15 Hrs.**
Unit IV	**Determinants and Matrices** Matrices, determinants, system of linear equations, eigenvalues and eigenvectors, LU decomposition.	**15 Hrs.**

Recommended Books

1. Hamdy A. Taha. "Operations Research: An Introduction". Pearson.
2. B.S. Grewal. "Higher Engineering Mathematics". Khanna Publishers.
3. J.L. Mott, A. Kandel and T.P. Baker: Discrete Mathematics for Computer Scientists, Reston, Virginia, 1983.
4. R.A. Brualdi: Introductory Combinatorics, North-Holland, New York, 1977.
5. W. Feller: An Introduction to Probability Theory and its Applications (Volume I and II), 3rd ed. John Wiley, New York, 1973

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Course Outcomes:
- Solve optimization problems using linear programming and simplex methods.
- Apply combinatorics, probability, and random variable distributions.
- Understand group theory and its applications in algebraic structures.
- Solve systems of equations and perform matrix operations, including eigenvalues.
- Develop problem-solving skills across various mathematical domains.

CS-MJ-T-13	**Artificial Intelligence**	Semester	Credit	Hours	Evaluation
Major		VII	4	40	15+60
Theory					

Course Objectives

This course introduces the basic principles of artificial intelligence, covering fundamental topics such as representation schemes, problem-solving paradigms, constraint propagation, and search strategies. It also explores key areas of AI applications, including knowledge representation, natural language processing, and expert systems, providing students with a broad understanding of AI concepts and techniques.

Syllabus

Unit I	**Introduction** Intelligent Agents – Agents and environments - Good behavior - The nature of environments- Structure of agents- Problem Solving - Problem solving agents- Example problems- Searching for solutions- Uniformed search strategies- Avoiding repeated states.	**6 Hrs.**
Unit II	**Searching Techniques** Informed search strategies- Heuristic function- Local search algorithms and optimistic problems- Local search in continuous spaces- Online search agents and unknown environments- Constraint satisfaction problems (CSP)- Backtracking search and Local search for CSP – Structure of problems- Adversarial Search – Games- Optimal decisions in games – Alpha – Beta Pruning.	**12 Hrs.**
Unit III	**Knowledge Representation** First order logic – Representation revisited- Syntax and semantics for first order logic Using first order logic-Knowledge engineering in first order logic- Inference in First order logic – prepositional versus first order logic- Unification and lifting- Forward chaining – backward chaining- Resolution.	**10 Hrs.**
Unit IV	**Planning and Reasoning with Uncertainty** Planning – classical planning and STRIPS representation, partial-order planning concepts, hierarchical task network planning, state-space and plan-space search approaches, Rule-Based Expert Systems – architecture and examples, Reasoning with Uncertainty – probability review and Bayesian reasoning, Reasoning with Uncertainty – Bayesian networks, Dempster–Shafer theory, certainty factors and expert system applications.	**12 Hrs.**

Recommended Books

1. Artificial Intelligence – A Modern Approach, Stuart Russell, Peter Norvig, 3rd Edition, Pearson Education, 2015.
2. Artificial Intelligence, Elaine Rich and Kevin Knight, 2nd Edition, Tata McGraw-Hill, 2003
3. Artificial Intelligence-Structures and Strategies for Complex Problem Solving, George F. Luger, Pearson Education 2002

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Explain the foundations and history of Artificial Intelligence, as well as the science of agent design.
- Illustrate the use of problem-solving techniques, such as the various search methods, games, and constraint satisfaction problems.
- Demonstrate AI's use of knowledge representation, through logic agents and first-order logic to address AI problems.
- Design simple software to experiment with various AI learning concepts and analyze results.
- Build self-learning and research skills to be able to tackle a topic of interest on his/her own or as part of a team.

CS-MJ-P-13	**Artificial Intelligence Lab**	Semester	Credit	Hours	Evaluation
Major		VII	2	30	60+10+5
Practical					

Suggested Programs

Introduction to Intelligent Agents

- Simulate different types of agents (simple reflex, model-based, goal-based, utility-based) using Python.
  Tools: Python, Jupyter Notebook, matplotlib for visualization.

Uninformed Search Techniques

- Implement **Breadth-First Search (BFS)**, **Depth-First Search (DFS)**, and **Uniform Cost Search** for maze/pathfinding problems.
  Tools: Python, networkx or custom graph classes.

Informed Search and Heuristics

- Implement **A*** and **Greedy Best-First Search** for solving puzzles (e.g., 8-puzzle, pathfinding).
  Tools: Python, heapq, numpy.

Local Search Algorithms

- Solve optimization problems using **Hill Climbing**, **Simulated Annealing**, and **Genetic Algorithms**.
  Tools: Python, random, matplotlib.

Online Search and Unknown Environments

- Build a simple grid world where an agent navigates without prior map knowledge.
  Tools: Python, pygame (optional for UI).

Constraint Satisfaction Problems (CSP)

- Solve classic CSPs like **Sudoku**, **Map Coloring**, or **N-Queens** using **Backtracking** and **Forward Checking**.
  Tools: Python, Jupyter Notebook.

Adversarial Search (Game Playing)

- Create a **Tic-Tac-Toe** or **Connect Four** agent using **Minimax** and **Alpha-Beta Pruning**.
  Tools: Python, tkinter (optional GUI).

First-Order Logic Representation

- Encode facts and rules in **First Order Logic (FOL)** and perform **Forward and Backward Chaining**.
  Tools: Prolog or Python-based logic engine (e.g., pyDatalog).

Resolution and Unification

- Demonstrate **Unification** of FOL statements and apply **Resolution** for theorem proving.
  Tools: Python, custom logic rule engine.

Lab Requirements

- **Software**: Python (Anaconda), Jupyter Notebook, scikit-learn, numpy, matplotlib, pygame (optional)
- **Datasets**: UCI repository (Iris, Wine, Titanic), custom puzzles and game environments.

Hardware: Standard desktop/laptop with 8GB RAM

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Implement and compare intelligent agents, search algorithms (uninformed, informed, local), and adversarial search techniques in Python.
- Solve constraint satisfaction and optimization problems using backtracking, CSP techniques, and metaheuristics.
- Model and reason with first-order logic using forward/backward chaining and resolution.
- Demonstrate practical AI concepts through simulations, puzzles, and game environments using Python tools and repositories.

CS-MJ-T-14	**Theory of Computation**	Semester	Credit	Hours	Evaluation
Major		VII	6	60	15+60
Theory					

Course Objectives

This course aims to provide a comprehensive understanding of the Theory of Computation, focusing on fundamental concepts and their applications. It explores finite state automata, including both deterministic and nondeterministic types, regular expressions, and their equivalence with finite state machines. The course also covers context-free grammars and pushdown automata, examining their properties, simplification techniques, and derivation trees. Additionally, it delves into Turing machines, their variants, the Chomsky hierarchy, and the theory of computability. Finally, it addresses decidability issues, such as Post's correspondence problem, to understand the limits of computation and the challenges within modern computing systems.

Syllabus

Unit I	**Language and Regular Grammer** Alphabets, string, language, basic operations on language, concatenation, union, Kleene star. Finite Automata (FA): Non deterministic FA and deterministic FA, NFA with ε- moves, Regular Expressions, Equivalence of regular expression and FA, Pumping lemma, closure properties, FA minimization: Myhill-Nerode theorem and Equivalence theorem, Formal Language and Grammar: Production systems, Finite automata with output, Regular grammar.	**15 Hrs.**
Unit II	**Context free Grammars** Context free grammars, Normal forms, Simplification of CFG, Derivation trees and ambiguity. Pushdown automata: Acceptance by empty store and final state, Deterministic pushdown automata, Equivalence between pushdown automata and context-free grammars, Pumping lemma for CFL, Closure properties of CFL.	**15 Hrs.**
Unit III	**Turing Machine** Turing machine as a model of computation, configuration of simple Turing machine, Church Turing Thesis, Universal Turing Machine, decidability, halting problem, context sensitive languages and linear bounded automata, Chomsky Hierarchy.	**15 Hrs.**
Unit IV	**Undecidability** A Language that is not Recursively Enumerable (RE), an Undecidable Problem that is RE, Undecidable Problems about Turing Machine. Post's Correspondence Problem, The Classes P and NP.	**15 Hrs.**

Recommended Books

1. Hopcraft, J. E., Motwani, R., & Ullman, J. D. (n.d.). Introduction to automata theory, languages and computation. Pearson.
2. Linz, P. An introduction to formal language and automata. Narosa Publisher.
3. Mishra, K. L. P., &Chandrasekaran, N. Theory of computer science: Automata, languages and computation. PHI.
4. Martin, T. C. Theory of computation. Tata McGraw-Hill.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Gain a deep understanding of finite automata, including deterministic and nondeterministic types, and their equivalence with regular expressions, as well as concepts like the Pumping Lemma and closure properties.
- Master context-free grammars, pushdown automata, and their relationships, including normal forms, derivation trees, and ambiguity.
- Achieve proficiency in Turing machines, covering their construction, variations, and the Chomsky hierarchy, as well as concepts of computability.
- Analyze decidability issues, such as Post's correspondence problem, equipping students with critical skills to address theoretical and practical challenges in computation.

CS-MJ-T-15	**Software Engineering**	Semester	Credit	Hours	Evaluation
Major		VII	6	60	15+60
Theory					

Course Objectives

This course covers key aspects of software engineering, including the role and characteristics of software, software process models, and CMMI. It focuses on requirement analysis, project management, risk management, and quality assurance. Topics also include design engineering, software architecture, and testing strategies such as black-box and white-box testing, offering a comprehensive understanding of the software development lifecycle.

Syllabus

Unit I	**Introduction** The Evolving Role of Software, Software Characteristics, Changing Nature of Software, Software Engineering as a Layered Technology, Software Process Framework, Framework and Umbrella Activities, Process Models, Capability Maturity Model Integration (CMMI).	**8 Hrs.**
Unit II	**Requirement Analysis** Software Requirement Analysis, Initiating Requirement Engineering Process, Requirement Analysis and Modelling Techniques, Flow Oriented Modelling, Need for SRS, Characteristics and Components of SRS.	**8 Hrs.**
Unit III	**Software Project Management** Estimation in Project Planning Process, Project Scheduling.	**8 Hrs.**
Unit IV	**Risk Management** Software Risks, Risk Identification, Risk Projection and Risk Refinement, RMMM Plan.	**8 Hrs.**
Unit V	**Quality Management** Quality Concepts, Software Quality Assurance, Software Reviews, Metrics for Process and Projects.	**8 Hrs.**
Unit VI	**Design Engineering** Design Concepts, Architectural Design Elements, Software Architecture, Data Design at the Architectural Level and Component Level, Mapping of Data Flow into Software Architecture, Modelling Component Level Design.	**10 Hrs.**
Unit VII	**Testing Strategies & Tactics** Software Testing Fundamentals, Strategic Approach to Software Testing, Test Strategies for Conventional Software, Validation Testing, System testing, Black-Box Testing, White-Box Testing and their type, Basis Path Testing.	**10 Hrs.**

Recommended Books

1. R.S. Pressman, Software Engineering: A Practitioner's Approach (7th Edition), McGrawHill, 2009.
2. P. Jalote, An Integrated Approach to Software Engineering (2nd Edition), Narosa Publishing House, 2003.
3. K.K. Aggarwal and Y. Singh, Software Engineering (2nd Edition), New Age International Publishers, 2008.
4. I. Sommerville, Software Engineering (8th edition), Addison Wesley, 2006.
5. D. Bell, Software Engineering for Students (4th Edition), Addison-Wesley, 2005.
6. R. Mall, Fundamentals of Software Engineering (2nd Edition), Prentice-Hall of India, 2004.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Understand the evolving role, characteristics, and process models of software development.
- Gain proficiency in requirement analysis, modeling, and the creation of Software Requirements Specifications (SRS).
- Develop skills in project management, estimation, scheduling, and risk management within software projects.
- Apply quality assurance practices and software testing strategies, including both black-box and white-box techniques.
- Learn to design software architectures and integrate data flow within component-level designs.

CS-MI-P-6	**Web Development and Applications**	Semester	Credit	Hours	Evaluation
Minor		VII	4	40	35+10+5
Practical					

Suggested Programs

- **Creating a Simple Web Page with HTML**
  Use basic tags: headings, paragraphs, bold, italic, lists.
- **Formatting and Organizing Content**
  Use tables, hyperlinks, images, and lists on a single webpage.
- **Designing a Contact Form in HTML**
  Include input fields, text areas, checkboxes, radio buttons, and a submit button.
- **Embedding Audio and Video in a Web Page**
  Add <audio> and <video> elements with controls and source formats.
- **Working with CSS for Styling**
  Apply inline, internal, and external CSS. Style text, backgrounds, and borders.
- **Building a Web Page Layout Using CSS**
  Use positioning techniques (relative, absolute, fixed), and the box model.
- **Designing a Multi-page Website**
  Use navigation links between pages (e.g., Home, About, Contact).
- **Introduction to JavaScript Programming**
  Basic arithmetic operations, alerts, and console logging.
- **Form Validation using JavaScript**
  Validate required fields, check for email format, and password length.
- **JavaScript Events and Interactivity**
  Implement onClick, onMouseOver, and onChange events with buttons and inputs.
- **Basic PHP Script**
  Create a script to display dynamic content using variables and control structures.
- **Working with PHP Forms**
  Retrieve and display form data using $_POST and $_GET methods.
- **PHP Arrays and Looping**
  Store and loop through data using indexed and associative arrays.
- **File Handling in PHP**
  Read from and write to a text file using PHP functions.
- **Introduction to MySQL and phpMyAdmin**
  Create a database and tables, insert sample data using phpMyAdmin.
- **Connecting PHP to a MySQL Database**
  Establish database connection using mysqli_connect() or PDO.
- **Displaying Data from Database using PHP**
  Fetch records using SELECT and display results in an HTML table.
- **Insert Form Data into MySQL Database**
  Capture form input and save it into a database using INSERT INTO.
- **Updating and Deleting Database Records**
  Implement update and delete operations using UPDATE and DELETE queries.
- **Building a Simple Login System**
  Use sessions, cookies, and basic user authentication using PHP and MySQL.

Suggested books

1. HTML5: The Missing Manual by Matthew MacDonald
2. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics by Jennifer Niederst Robbins
3. CSS Secrets: Better Solutions to Everyday Web Design Problems by Lea Verou
4. HTML and CSS: Visual QuickStart Guide by Elizabeth Castro
5. Eloquent JavaScript: A Modern Introduction to Programming by Marijn Haverbeke
6. Learning MySQL: Get a Handle on Your Data by Seyed M.M. & Jason Geck
7. PHP and MySQL for Dynamic Web Sites by Larry Ullman

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design static and dynamic web pages using HTML, CSS, and JavaScript, including responsive layouts, multimedia embedding, and client-side form validation.
- Develop interactive, multi-page websites with navigation, styling, and user input handling using both front-end and back-end (PHP) technologies.
- Implement database-driven web applications: create, read, update, and delete records in MySQL using PHP, and build a basic login system with session management.
- Demonstrate end-to-end web development skills by integrating forms, databases, and server-side logic for functional, user-friendly websites.

Detail Syllabus of Semester-VIII

CS-MJ-T-16	**Machine Learning**	Semester	Credit	Hours	Evaluation
Major		VIII		40	15+60
Theory					

Course Objectives

The course will provide students with a comprehensive foundation in machine learning, introducing key concepts and principles such as supervised, unsupervised, and reinforcement learning. Students will gain a deep understanding of various machine learning algorithms, learning when and how to apply them effectively. Practical skills will be developed through hands-on implementation and evaluation of machine learning models using popular programming languages and libraries. Additionally, the course will cover essential data handling techniques, including data preprocessing, feature engineering, and data visualization. Ethical considerations, such as fairness, bias, and the societal impacts of machine learning models, will also be discussed, ensuring students are aware of the broader implications of their work.

Syllabus

Unit I	**Supervised Learning (Regression/Classification)** Overview for ML, Supervised Learning, Unsupervised Learning, Problems, Data, and Tools. Linear Regression, Polynomial Regression, Features, Scaling, Cost Function, Gradient Descent, Learning Rate. Distance- based methods, Nearest-Neighbours, Decision Trees, Naïve Bayes, Linear Regression, Logistic Regression, Generalized Linear Models-Support Vector Machines, Nonlinearity and Kernel Multi-class/Structured Outputs, Ranking.	**10 Hrs.**
Unit II	**Unsupervised Learning** K-means Clustering, PCA and kernel PCA -Matrix Factorization and Matrix Completion Generative Models (mixture models and latent factor models), Expectation Maximization, Dimensionality Reduction.	**10 Hrs.**
Unit III	**Performance Evaluation** Evaluating Machine Learning algorithms and Model Selection, Performance Measure, Error Analysis, Confusion Matrix, Precision and Recall Tradeoff, F1 Score, Macro F1, Accuracy, Skewed Classes Introduction to Statistical Learning Theory, Ensemble Methods (Boosting, Bagging, Random Forests).	**10 Hrs.**
Unit IV	**Modelling and Applications** Sparse Modeling and Estimation, Modeling Sequence/Time-Series Data, Feature Representation Learning. Discriminative vs. Generative Models, Marginalization, Conditioning, Normalization, and Conditional Independence, Bayes Theorem, Markov Random Field, Naïve Bayes Model, Decision Tree, Recent trends in various learning techniques of machine learning and classification methods for IOT applications.	**10 Hrs.**

Recommended Books

1. Tom Mitchell, "Machine Learning", McGraw-Hill, 1997.
2. Kevin Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012
3. Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning, Springer 2009
4. Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2007.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Conceptual Mastery: Students will understand key concepts and techniques in machine learning, including different types of learning and model evaluation metrics
- Algorithm Application: Students will be able to select and apply suitable machine learning algorithms to solve specific problems.
- Implementation Skills: Students will be proficient in using tools such as Python, scikit-learn, TensorFlow, or PyTorch to build and evaluate machine learning models.
- Data Proficiency: Students will demonstrate the ability to preprocess and visualize data, and perform feature selection and extraction.
- Ethical Awareness: Students will recognize and address ethical issues and biases in machine learning, ensuring responsible and fair use of technology.

CS-MI-P-16	**Machine Learning Lab**	Semester	Credit	Hours	Evaluation
Major		VIII	2	30	60+10+5
Practical					

Suggested Programs

1. **Supervised Learning (Regression/Classification)**
   - **Implementing Linear Regression**: Build a simple linear regression model from scratch and evaluate its performance on a dataset.
   - **Polynomial Regression**: Implement polynomial regression and compare it with linear regression for non-linear data.
   - **Gradient Descent**: Implement gradient descent for optimizing linear regression and evaluate the effect of learning rates on convergence.
   - **K-Nearest Neighbors (K-NN)**: Implement the K-NN algorithm for classification and explore different distance metrics.
   - **Decision Trees**: Implement a decision tree classifier for a classification problem and analyze its performance.
   - **Logistic Regression**: Implement logistic regression for binary classification and discuss the use of sigmoid functions.
   - **Support Vector Machines (SVM)**: Implement SVM with linear and non-linear kernels and compare its performance to other classifiers.
   - **Naïve Bayes**: Implement a Naïve Bayes classifier for a text classification task (e.g., spam vs. non-spam).
   - **Multiclass Classification**: Apply one-vs-all and one-vs-one approaches for multi-class classification using algorithms like logistic regression or SVM.
2. **Unsupervised Learning**
   - **K-means Clustering**: Implement the K-means clustering algorithm and visualize the results on a dataset.
   - **Principal Component Analysis (PCA)**: Apply PCA for dimensionality reduction and visualize the data in 2D.
   - **Kernel PCA**: Implement Kernel PCA and compare it with regular PCA for non-linear datasets.
   - **Expectation Maximization (EM)**: Implement the EM algorithm for Gaussian Mixture Models (GMM) and use it for clustering.
   - **Dimensionality Reduction**: Explore different dimensionality reduction techniques (e.g., t-SNE) for high-dimensional datasets.
3. **Performance Evaluation**
   - **Confusion Matrix**: Implement a confusion matrix and calculate performance metrics such as accuracy, precision, recall, and F1 score.
   - **Cross-Validation**: Implement K-fold cross-validation for model selection and evaluation.
   - **Error Analysis**: Perform error analysis for classification algorithms and discuss the trade-offs between precision and recall.
   - **ROC and AUC**: Implement ROC curves and compute the AUC for binary classification problems.
   - **Ensemble Methods (Bagging and Boosting)**: Implement random forests (bagging) and boosting algorithms such as AdaBoost or XGBoost for model improvement.
   - **Boosting and Bagging**: Compare the performance of boosting and bagging methods using decision trees as base learners..
4. **Modelling and Applications**
   - **Time-Series Modeling**: Implement and evaluate time-series forecasting models (e.g., ARIMA, LSTM) on a sequence dataset.
   - **Sparse Modeling**: Implement L1 regularization (Lasso) for sparse feature selection and model estimation.
   - **Bayesian Inference**: Implement a basic Bayesian model (e.g., Naïve Bayes) and apply Bayes Theorem to a classification problem.
   - **Hidden Markov Models (HMM)**: Implement an HMM for sequence data prediction (e.g., speech recognition or part-of-speech tagging).
   - **Generative vs. Discriminative Models**: Compare and contrast generative models (e.g., Naïve Bayes) with discriminative models (e.g., Logistic Regression).

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Implement and evaluate supervised (regression, classification) and unsupervised (clustering, dimensionality reduction) learning algorithms from scratch in Python.
- Analyze model performance using metrics, cross-validation, error analysis, and visualization (ROC, confusion matrix).
- Apply ensemble methods (bagging, boosting), time-series models, Bayesian inference, and generative/discriminative approaches to real datasets.
- Compare algorithmic approaches for different problem types, exploring their trade-offs, strengths, and practical applications.

CS-MJ-T-17	**Soft Computing**	Semester	Credit	Hours	Evaluation
Major		VIII	6	60	15+60
Theory					

Course Objectives

This course introduces soft computing techniques and their applications, providing a solid foundation in fuzzy logic, neural networks, and evolutionary algorithms. It enhances students' problem-solving skills by teaching them how to apply these methods to complex real-world problems. The course emphasizes the interdisciplinary nature of soft computing, showing how it integrates with various fields, and equips students with the practical skills necessary to implement and experiment with soft computing techniques across different domains.

Syllabus

Unit I	**Introduction to Soft Computing** Introduction to soft computing; introduction to biological and artificial neural network; introduction to fuzzy sets and fuzzy logic systems.	**14 Hrs.**
Unit II	**An Introduction to Artificial Neural Network** Fundamental concepts, Evolution of NN, Basic Models of ANN, connections and learning, Terminologies such as weights, Bias, Threshold, Learning Rate etc., McCulloch-Pitts Neuron, Heb Network; Supervised Learning Network: Perceptron Network, Adaptive Linear Neuron, Multiple Adaptive Linear Neurons, Backpropagation Network, Radial Basis Function Network; Associate Memory Networks: Introduction and training algorithm for pattern association, Auto-associative Memory Network, Hetro-associative Memory Network, Bidirectional associative memory, Hopfield Network; Unsupervised Learning Network: Introduction; Fixed Weight Competitive Nets; Kohonen Self-Organizing Feature Maps; Adaptive Resonance Theory; Applications of ANN: Applications: such as recognition of characters, fabric defect identification etc.	**16 Hrs.**
Unit III	**Introduction to Fuzzy Logic** Classical Sets, Fuzzy Sets: operations and properties. Operations on fuzzy relations; Membership functions: Features, fuzzification, methods of membership value assignments; Defuzzification: Introduction; Lambda-Cuts for fuzzy sets and fuzzy relations; Defuzzification methods; Fuzzy Rules: Introduction; formation of rules, decomposition and aggregation of rules; fuzzy reasoning; Fuzzy inference systems (FIS) and applications: FIS methods: Mamdani and Sugeno; Applications: such as fuzzy logic control etc.	**15 Hrs.**
Unit IV	**Genetic Algorithms** Concept of Genetics and Evolution and its application to probabilistic search techniques, Basic GA framework and different GA architectures, GA operators: Encoding, Crossover, Selection, Mutation, etc., Solving single objective optimization problems using GAs.	**15 Hrs.**

Recommended Books

1. Timothy J. Ross, "Fuzzy Logic with Engineering Applications," McGraw Hill, 1995.
2. Simon Haykin, "Neural Networks" Pearson Education.
3. B. Yegnanarayana, "Artificial Neural Networks," PHI, India, 2006.
4. S. N. Sivanandan and S. N. Deepa, "Principles of Soft Computing", Wiley India, 2012.
5. Fakhreddine O. Karray and Clarence De Silva., "Soft Computing and Intelligent Systems Design, Theory, Tools and Applications," Pearson Education, India, 2009.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Conceptual Clarity: Students will understand the fundamental principles of soft computing techniques such as fuzzy logic, neural networks, and genetic algorithms.
- Analytical Skills: Students will be able to analyze and design solutions for complex problems using soft computing methods.
- Implementation Skills: Students will gain proficiency in implementing soft computing algorithms and models using appropriate software tools.
- Application Insight: Students will demonstrate the ability to apply soft computing techniques to real-world problems in areas like optimization, control systems, and pattern recognition.
- Interdisciplinary Approach: Students will appreciate the interdisciplinary nature of soft computing and its potential to innovate across various fields.

CS-MJ-T-18	**Data Warehousing and Data Mining**	Semester	Credit	Hours	Evaluation
Major		VIII	4	40	15+60
Theory					

Course Objectives

The objective of this course is to equip students with a comprehensive understanding of data warehousing principles, business intelligence tools, and advanced data mining techniques. Students will learn to design, build, and manage data warehouses, perform multidimensional data analysis, and apply mining techniques to extract meaningful patterns and insights from large-scale datasets.

Syllabus

Unit I	**Data Warehousing and Business Analysis** Data warehousing Components –Building a Data warehouse –Data Warehouse Architecture – DBMS Schemas for Decision Support – Data Extraction, Cleanup, and Transformation Tools –Metadata – reporting – Query tools and Applications – Online Analytical Processing (OLAP) – OLAP and Multidimensional Data Analysis.	**4 Hrs.**
Unit II	**Data Mining** Data Mining Functionalities – Data Preprocessing – Data Cleaning – Data Integration and Transformation – Data Reduction – Data Discretization and Concept Hierarchy Generation- Architecture Of A Typical Data Mining Systems- Classification Of Data Mining Systems.	**8 Hrs.**
Unit III	**Association Rule Mining** Efficient and Scalable Frequent Item set Mining Methods – Mining Various Kinds of Association Rules – Association Mining to Correlation Analysis – Constraint-Based Association Mining.	**8 Hrs.**
Unit IV	**Cluster Analysis** Types of Data in Cluster Analysis – A Categorization of Major Clustering Methods – Partitioning Methods – Hierarchical methods – Density-Based Methods – Grid-Based Methods – Model Based Clustering Methods – Clustering High-Dimensional Data – Constraint-Based Cluster Analysis – Outlier Analysis.	**10 Hrs.**
Unit V	**Classification and Prediction** Issues Regarding Classification and Prediction – Nearest Neighbor Classification - Classification by Decision Tree Introduction – Bayesian Classification – Rule Based Classification –Support Vector Machines – Associative Classification –Prediction – Accuracy and Error Measures – Evaluating the Accuracy of a Classifier or Predictor – Ensemble Methods – Model Section. Mining Object, Spatial, Multimedia, Text & Web Data. Multidimensional Analysis and Descriptive Mining of Complex Data Objects – Spatial Data Mining – Multimedia Data Mining – Text Mining – Mining the World Wide Web.	**10 Hrs.**

Recommended Books

1. Reema Thareja, “Data Warehousing”, Oxford University Press.
2. Jiawei Han and Micheline Kamber, “Data Mining Concepts & Techniques”, Elsevier Pub.
3. Margret H. Dunham “Data Mining: Introductory and Advanced topics” Pearson Education.
4. Paulraj Ponniah, “Data Warehousing Fundamentals”, John Wiley & Sons, Inc.
5. Vikram Pudi, P. Radha Krishana “Data Mining”, Oxford University press.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Design and implement data warehouses using appropriate schemas (Star, Snowflake) and perform ETL processes effectively.
- Apply OLAP operations and multidimensional analysis for business intelligence and decision support.
- Utilize data mining techniques (classification, clustering, association rules) to discover hidden patterns in large datasets.
- Preprocess and prepare data using cleaning, transformation, and reduction techniques for analytical tasks.
- Analyze and mine complex data such as spatial, text, and web data for real-world business applications.

CS-MJ-P-18	**Data Warehousing and Data Mining Lab**	Semester	Credit	Hours	Evaluation
Major		VIII	2	30	60+10+5
Practical					

Course Objectives

The course aims to equip students with theoretical and practical knowledge of data warehousing, data mining, and business analytics techniques. It focuses on data preprocessing, OLAP, association rule mining, clustering, classification, and the analysis of complex data types, enabling students to design, implement, and analyze intelligent decision-support systems for real-world applications.

Suggested Programs / Lab Experiments

Design and Create a Data Warehouse

Implement star schema and snowflake schema using a sample dataset.

ETL (Extract, Transform, Load) Implementation

Use ETL tools (e.g., Talend, Pentaho, SSIS) to load cleaned and transformed data into the warehouse.

Metadata Creation and Management

Store and query metadata in the data warehouse environment.

OLAP Cube Design and Querying

Create and query OLAP cubes for multidimensional data analysis.

Data Preprocessing

Perform data cleaning, integration, transformation, and reduction using Python or R.

Association Rule Mining

Apply Apriori and FP-Growth algorithms on transactional datasets to discover frequent itemsets.

Correlation and Constraint-Based Association Mining

Analyze strong and weak correlations in datasets using association mining techniques.

Clustering Analysis

Implement k-Means, hierarchical clustering, and DBSCAN using sample datasets.

Classification Models

Implement Decision Tree, Naïve Bayes, and k-Nearest Neighbor classifiers.

Prediction and Model Evaluation

Build regression models and evaluate accuracy using confusion matrix, precision, recall, and F1-score.

Ensemble Methods in Classification

Apply Bagging, Boosting, and Random Forest methods to improve prediction accuracy.

Text Mining

Extract keywords, perform sentiment analysis, and generate word clouds from text datasets.

Web Mining

Scrape and analyze web data for trend detection.

Spatial and Multimedia Data Mining

Use GIS datasets for spatial analysis; extract features from images or audio datasets.

Mini Project

End-to-end data warehousing and mining project integrating ETL, OLAP, and advanced analytics.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Understand the architecture, components, and design principles of data warehouses and data mining systems.
- Apply data preprocessing, ETL, and OLAP techniques for decision-support applications.
- Analyze datasets to discover patterns using association rule mining, clustering, and classification algorithms.
- Evaluate the performance of mining models using accuracy metrics and select appropriate methods for specific business problems.
- Create end-to-end data warehouse and business analytics solutions integrating multiple analytical techniques.

CS-MJ-T-19	**Computer Graphics**	Semester	Credit	Hours	Evaluation
Major		VIII	4	60	15+60
Theory					

Course Objectives

The objective of this course is to provide students with a comprehensive understanding of the fundamental concepts, algorithms, and techniques of computer graphics. Students will learn about graphics systems and devices, scan conversion algorithms, two-dimensional and three-dimensional transformations, viewing and clipping, curve representation, hidden surface removal, and color and shading models. The course also aims to develop the ability to apply computer graphics techniques for visualization, image representation, and graphical applications.

Syllabus

Unit I	**Introduction to Computer Graphics and Graphics Systems** Overview of computer graphics – Representing pictures – Preparing, presenting, and interacting with pictures for presentations – Visualization and image processing – RGB color model – Direct coding – Lookup table – Storage tube graphics display – Raster scan display – 3D viewing devices – Plotters – Printers – Digitizers – Light pens – Active and passive graphics devices – Computer graphics software.	**10 Hrs.**
Unit II	**Scan Conversion** Points and lines – Line drawing algorithms – DDA algorithm – Bresenham's line algorithm – Circle generation algorithm – Ellipse generation algorithm – Scan-line polygon fill algorithm – Boundary fill algorithm – Flood fill algorithm.	**10 Hrs.**
Unit III	**2D Transformation and Viewing** Basic transformations: translation, rotation, and scaling – Matrix representations and homogeneous coordinates – Transformations between coordinate systems – Reflection – Shear – Transformation of points, lines, parallel lines, and intersecting lines – Viewing pipeline – Window-to-viewport coordinate transformation – Clipping operations – Point clipping – Line clipping – Clipping of circles, polygons, and ellipses.	**16 Hrs.**
Unit IV	**3D Transformation and Viewing** 3D transformations: translation, rotation, scaling, and other transformations – Rotation about an arbitrary axis in space – Reflection through an arbitrary plane – General parallel projection transformation – Clipping – Viewport clipping – 3D viewing.	**14 Hrs.**
Unit V	**Curves, Hidden Surfaces, Color and Shading Models** Curves: Curve representation – Surfaces – Designs – Bezier curves – B-spline curves – End conditions for periodic B-spline curves – Rational B-spline curves. Hidden Surfaces: Depth comparison – Z-buffer algorithm – Back-face detection – BSP tree method – Painter's algorithm – Scan-line algorithm – Hidden-line elimination – Wire-frame methods – Fractal geometry. Color and Shading Models: Light and color model – Interpolative shading model – Texture.	**10 Hrs.**

Recommended Books

1. J. D. Foley, A. Van Dam, S. K. Feiner and J. F. Hughes, Computer Graphics: Principles and Practice, 2nd Edition, Addison-Wesley, 1990.
2. D. Hearn and M. P. Baker, Computer Graphics, Prentice Hall of India, 2008.
3. D. F. Rogers, Procedural Elements for Computer Graphics, McGraw-Hill, 1997.
4. D. F. Rogers and J. A. Adams, Mathematical Elements for Computer Graphics, 2nd Edition, McGraw-Hill, 1989.
5. D. Hearn and M. P. Baker, Computer Graphics with OpenGL, Pearson Education.
6. Arup Mukherjee, Fundamentals of Computer Graphics & Multimedia, PHI.
7. Arup Mukherjee, Introduction to Computer Graphics, Vikas Publishing House.

Outcome

After completing this course, students are expected to demonstrate the following knowledge, skills, and abilities:

- Explain the fundamental concepts of computer graphics, graphics systems, display devices, input devices, and graphics software.
- Apply scan conversion algorithms for generating lines, circles, ellipses, and filling polygons.
- Apply two-dimensional and three-dimensional transformations, viewing techniques, and clipping algorithms.
- Represent and manipulate curves and surfaces using Bezier, B-spline, and rational B-spline techniques.
- Apply hidden surface removal, wire-frame, color, shading, and texture techniques in computer graphics applications.