GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

(Autonomous)

(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)

Madhurawada :: Visakhapatnam – 530 048

COURSE STRUCTURE

(Applicable for the Academic Year 2024-25 onwards)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.Tech. Computer Science and Engineering

SCHEME AND SYLLABI

(With effect from 2024-25 admitted batch)

II Year - I Semester

Course Code	Category	Course Title	Hours per Week		Internal Marks	External Marks	Total Marks	Credits
			L	P				
24BM11RC05	ES	Discrete Mathematical Structures	3	0	30	70	100	3
24CT11RC11	PC	Computer Organization	3	0	30	70	100	3
24CT11RC09	PC	Data Structures	3	0	30	70	100	3
24CT11RC10	PC	Object Oriented Programming Through Java	3	0	30	70	100	3
24CT11RC13	PC	Operating Systems	3	0	30	70	100	3
24CT11RC14	PC	Data Structures Lab	0	3	50	50	100	1.5
24CT11RC15	PC	Object Oriented Programming Through Java Lab	0	3	50	50	100	1.5
24CT11RC17	PC	Operating Systems Lab	0	3	50	50	100	1.5
24CS11SC01	SC	Skill Course-1: Advanced Python Programming	1	2	50	50	100	2
24BC11MC01	MC	Environment Science	2	0	50	50	100	0
							Total Credits	**21.5**

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-II

COMPUTER ORGANIZATION

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC11 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end the Course, the student shall be able to of	
CO1:	Demonstrate the design of major components of a basic computer such as register, transfer Micro-operations and arithmetic logic shift unit. (L2)
CO2:	Classify various instruction codes, timing & control unit and understand the design of CPU, Micro-programmed control Unit. (L2)
CO3:	Make Use of register organization, instruction formats, and addressing modes to understand the working of central processing units (CPUs) and 8086 microprocessors. (L3)
CO4:	Interpret different ways of communication with I/O devices and parallel processors. (L2)
CO5:	Outline the performance of various memory modules and to comprehend computer arithmetic algorithms. (L2).

UNIT-I	**8 Lectures**
Register Transfer and Micro operations: Register Transfer Language, Register Transfer, Bus and Memory Transfers, Arithmetic Micro operations, Logic Micro-operations, Shift Micro operations, Arithmetic Logic Shift Unit.	

UNIT-II	**10 Lectures**
Basic Computer Organization and Design: Instruction Codes, Computer Registers, Computer Instructions, Timing and Control, Instruction Cycle, Memory-Reference Instructions, Input-Output and Interrupt, Complete Computer Description, Design of Basic Computer, Design of Accumulator Logic. **Micro Programmed Control:** Control Memory, Address Sequencing, Micro program Example, Design of Control Unit.	

UNIT-III	**12 Lectures**
Central Processing Unit: Introduction, General Register Organization, Stack Organization, Instruction Formats, Addressing Modes. Data Transfer and Manipulation, Program Control, Reduced Instruction Set Computer (RISC). **8086 Microprocessor:** Family Overview, Internal Architecture.	

UNIT-IV	**10 Lectures**
Pipeline and Vector Processing: Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction Pipeline. **Input/Output Organization:** Peripheral Devices, I/O interface, Asynchronous data transfer, Modes of transfer, priority Interrupt, Direct memory access.	

UNIT-V	**10 Lectures**
Computer Arithmetic: Addition and Subtraction with Signed-Magnitude Data, Booth Multiplicatior Algorithm. **Memory Organization:** Memory Hierarchy, Main memory, Associative Memory, Cache Memory.	

Text Books:
1. Computer System Architecture, M. Morris Mano, Prentice Hall of India Pvt. Ltd., Third Edition, Sept.2008. 2. Microprocessors and Interfacing, D. V. Hall, TMGH, 2nd Edition 2006.
Reference Books:
1. Computer Architecture and Organization, William Stallings, PHI Pvt. Ltd., Eastern Economy Edition, Sixth Edition, 2003.Applications of Linear ICs- Clayton, Publisher: Macmillan Education UK, Year: 1978. 2. Computer Organization and Architecture, Linda Null, Julia Lobur, Narosa Publications ISBN81- 7319-609-5. 3. Computer System Architecture‖, John. P.Hayes.
Web References:
1. NPTEL :: Computer Science and Engineering - NOC:Computer architecture and

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-III

DATA STRUCTURES

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT, ECE]

Course Code: 24CT11RC09 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Outline the concept of ADT, identify suitable data structures to solve problems, and experiment with different searching & sorting techniques using arrays. (L2, L3)
CO2:	Develop and analyse algorithms for stacks, queues, and priority queues. (L3, L4)
CO3:	Model linked list representations for various applications. (L3)
CO4:	Develop and analyse algorithms for Binary Trees and Binary Search Trees. (L3, L4)
CO5:	Summarize the concepts of Graph representation, graph traversals and hashing. (L2)

UNIT-I	**10 Lectures**
Data Structures: Definition, Classification of Data Structures, Operations on Data Structures, Abstract Data Type (ADT), Introduction to array ADT. **Searching:** Linear search, Binary search. **Sorting:** Insertion sort, Selection sort, Exchange (Bubble sort, quick sort), distribution (radix sort).	

UNIT-II	**10 Lectures**
Stacks: Introduction to Stacks, Array Representation of Stacks, Operations on Stacks, Applications-Infix to Postfix Conversion, Evaluating Postfix Expressions. **Queues:** Introduction to Queues, Representation of Queues-using Arrays, Implementation of Queues-using Arrays, Circular Queues. **Priority Queue:** model, simple implementation, Binary Heap-structure property, heap order property, heap operations, Heap sort.	

UNIT-III	**12 Lectures**
Linked List: Introduction, Single linked list, Representation of Linked list in memory, Operations on Single Linked list-Insertion, Deletion, Search and Traversal, Reversing Single Linked list, Double Linked list, Circular Linked list. **Applications on Single Linked list:** Stacks and queues using linked list, Polynomial Expression Representation, Sparse Matrix Representation using Linked List.	

UNIT-IV	**10 Lectures**
Introduction: Terminology, Representation of trees **Binary Trees:** The ADT, Properties of binary trees, Binary tree Representations, Binary tree Traversals: in order traversal, preorder traversal, post order traversal. **Binary Search Trees:** Definition, Searching a BST, Insertion into a BST, Deletion from a BST. **Efficient Binary Search Tree:** AVL Trees.	

UNIT-V	**8 Lectures**
Graphs: Basic Concepts, Representations of Graphs-Adjacency Matrix and using Linked list, Graph Traversals (BFT & DFT), Hashing-Introduction, static hashing, hashing functions, overflow handling techniques.	

Text Books:
1. Fundamentals of Data Structures in C, 2nd Edition, Horowitz, Sahin, Universities Press. 2. Data Structures and algorithm analysis in C, 2nded, Mark Allen Weiss.
Reference Books:
1. Data Structures Using C. 2nd Edition, ReemaThareja, Oxford. 2. Data Structures: A PseudoCode Approach, 2/e, Richard F.Gilberg, Behrouz A. Forouzon, Cengage.
Web References:
1. https://onlinecourses.swayam2.ac.in/cec25_ma15/preview.

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-IV

OBJECT ORIENTED PROGRAMMING THROUGH JAVA

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC10 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Explain the fundamental concepts of Object-Oriented Programming and Java features. (L2)
CO2:	Apply inheritance and polymorphism concepts to develop reusable and maintainable java applications. (L3)
CO3:	Utilize exception handling and multithreading techniques to enhance application reliability. (L3)
CO4:	Build interactive GUI applications using Java Swing and event-driven programming. (L3)
CO5:	Construct data handling mechanisms using Java I/O and develop JDBC applications to perform CRUD operations with MySQL. (L3)

UNIT-I	**12 Lectures**
An Overview of Java: Object-Oriented Programming- OOP Principles, The Java Buzzwords, A First Simple Program, Lexical Issues, Data Types, Variables, Arrays, Operators, Control Statements. **Introducing Classes**: Class Fundamentals, Declaring Objects, Assigning Object Reference Variables, A Closer Look at Methods and Classes - Introducing Methods, Constructors, this Keyword, Garbage Collection, Overloading Methods, Using Objects as Parameters, Understanding static, Introducing final, Introducing Nested and Inner Classes, Using Command-Line Arguments.	

UNIT-II	**10 Lectures**
Inheritance and Polymorphism: Inheritance Basics, Using super, Creating a Multilevel Hierarchy, Constructor invocation- Execution of Constructors, Method Overriding, Run-time polymorphism - Dynamic Method Dispatch, Using Abstract Classes, Using final with Inheritance, The Object Class. **Packages and Interfaces** –Packages and Member Access, Importing Packages, Interfaces, Use static Methods, String Handling.	

UNIT-III	**10 Lectures**
Exception Handling: Exception-Handling Fundamentals, Exception Types, Using try and catch, Multiple catch Clauses, Nested try Statements, Throw, Throws, Finally, Java's Built-in Exceptions, Creating Your Own Exception Subclasses, Chained Exceptions. **Multithreaded Programming:** The Java Thread Model, Creating a Thread and Multiple Threads, Thread States, Thread Properties, Thread Priorities, Synchronization, Inter thread Communication.	

UNIT-IV	**10 Lectures**
Applets and Event Handling: Applets- A Simple Applet, The Applet HTML tag and its Attributes, Two Event Handling Mechanisms, Delegation Event Model, Event Classes, Sources of Events, Event Listener Interfaces, Handling Mouse Events, Handling Keyboard Events, Adapter and Inner Classes. **Introducing GUI Programming:** Using AWT Controls, Layout Managers, and Menus, Introducing Swing.	

UNIT-V	**8 Lectures**
Input/Output: Exploring java.io, The Stream Classes, The Byte Streams, The Character Streams, Serialization. **Java Database Connectivity (JDBC):** JDBC Introduction, JDBC Architecture, Java Database Connectivity with MySQL, CRUD operations.	

Text Books:

1. Java The Complete Reference (11th ed.), Herbert Schildt (2018), McGraw Hill. ISBN: 978-1260440232.
2. Object-Oriented Programming with Java (8th ed.), Balagurusamy, E (2019). McGraw Hill.

Reference Books:

1. Core Java Volume I – Fundamentals (8th ed.), Horstmann, Gary Cornell (2018), Pearson. ISBN: 978-813171945-9.
2. Java How to Program, Early Objects Version (11th ed.), Deitel, P., & Deitel, H. (2017). Pearson.

Web References:

1. https://onlinecourses.nptel.ac.in/noc20_cs58/preview
2. https://docs.oracle.com/javase/tutorial/
3. https://www.udemy.com/course/java-tutorial/
4. https://www.edx.org/learn/java/universidad-carlos-iii-de-madrid-introduction-to-java-programming-starting-to-code-in-java
5. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-V

OPERATING SYSTEMS

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC13 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Classify operating system types, concepts, and core services to understand their role in managing hardware and software resources. (L2)
CO2:	Summarize the impact of scheduling policies and interprocess communication on system performance in multitasking environments. (L2)
CO3:	Make use of synchronization mechanisms and deadlock handling strategies to ensure safe and efficient concurrent process execution. (L3)
CO4:	Apply memory management techniques like paging, segmentation, and virtual memory to optimize resource utilization and system performance. (L3)
CO5:	Utilize file system structures, allocation methods, and disk scheduling algorithms to enhance data storage and retrieval efficiency. (L3)

UNIT-I	**10 Lectures**
Introduction: Operating System's role, Operating-System Structure, Operating-System Operations, Process Management, Memory Management, Storage Management, Protection and Security. **System Structures:** Operating-System Services, System Calls, Types of System Calls, Operating System Structure, Operating-System Generation, System Boot.	

UNIT-II	**10 Lectures**
Process Concept: Process Concept, Process Scheduling, Operations on Processes, Interprocess Communication. **Multithreaded Programming:** Overview, Multithreading Models, Threading Issues. **Process Scheduling**: Basic Concepts, Scheduling Criteria, Scheduling Algorithms, Thread Scheduling, Multiple Processor Scheduling.	

UNIT-III	**12 Lectures**
Synchronization: The Critical-Section Problem, Peterson's Solution, Synchronization Hardware, Mutex Locks, Semaphores, Classic Problems of Synchronization, Monitors. **Deadlocks**: System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock.	

UNIT-IV	**8 Lectures**
Memory-Management Strategies: Swapping, Contiguous Memory Allocation, Segmentation, Paging, Structure of the Page Table. **Virtual-Memory Management:** Demand Paging, Copy-on-Write, Page Replacement, Allocation of Frames, Thrashing.	

UNIT-V	**10 Lectures**
File System: File Concept, Access Methods, Directory and Disk Structure, File-System Mounting, File Sharing. **Implementing File-Systems**: File-System Structure, File-System Implementation, Directory Implementation, Allocation Methods, Free-Space Management. **Mass-Storage Structure:** Overview of Mass-Storage Structure, Disk Structure, Disk Scheduling, Swap-Space Management, RAID Structure.	

Text Books:

1. Operating System Concepts, Abraham Silberschatz, Peter B. Galvin, Greg Gagne, 9th Edition, Wiley, 2018, ISBN: 978-1118063330.
2. Operating Systems: Internals and Design Principles, William Stallings, 9th Edition, Pearson, 2018, ISBN: 978-1292214290

Reference Books:

1. Modern Operating Systems, Andrew S. Tanenbaum, Herbert Bos, 5th Edition, Pearson, 2022.
2. Operating Systems: A Concept-Based Approach, Dhananjay M. Dhamdhere, 3rd Edition, McGraw Hill, 2017.
3. Operating Systems: A Design-Oriented Approach, Charles Crowley, 1st Edition, McGraw Hill, 2019.

Web References:

1. https://nptel.ac.in/courses/106106144.
2. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-828-operating- system-engineering-fall-2012/.
3. https://cs140.stanford.edu

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-VI

DATA STRUCTURES LAB

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT, ECE]

Course Code: 24CT11RC14 **L-T-P-C: 0-0-3-1.5**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Apply different Searching and Sorting Techniques using arrays. (L3)
CO2:	Experiment with different linear data structure concepts using stacks and Queues. (L3)
CO3:	Develop linear data structure models using various Linked lists. (L3)
CO4:	Build Binary Search Tree & AVL tree and examine their traversals. (L3)
CO5:	Apply DFS and BFS graph traversal techniques. (L3)

List of Experiments

Implement the following programs with either C/C++/JAVA/Python

Module-1: Searching

1. Write a program that use non recursive functions to perform linear search for a Key value in a given list.
2. Write a program that use non recursive functions to perform Binary search for a Key value in a given list.

Module-2: Sorting

1. Write a program that implement Bubble sort, to sort a given list of integers in ascending order.
2. Write a program that implement Selection sort, to sort a given list of integers in ascending order.
3. Write a program that implement Insertion sort, to sort a given list of integers in ascending order.

Module-3: Efficient Sorting

1. Write a program that implement Quick sort, to sort a given list of integers in ascending order.

Module-4: Stack & Queue

1. Write a program that implement stack (its operations) using arrays.

2. Write a program that implement Queue (its operations) using arrays.

Module-5: Singly Linked List

1. Write a program that uses functions to create and perform operations on singly linked list.

Module-6: Double Linked List

1. Write a program that uses functions to create and perform operations on double linked list.

Module-7: Circular Linked List

1. Write a program that uses functions to create and perform operations on circular linked list.

Module-8: Binary Search Trees

1. Write a program to Create a Binary Search Tree and Perform insertion and deletion operations.

Module 9: AVL tree

1. Write a program to Build an AVL tree and perform insertions.

Module-10: Graphs

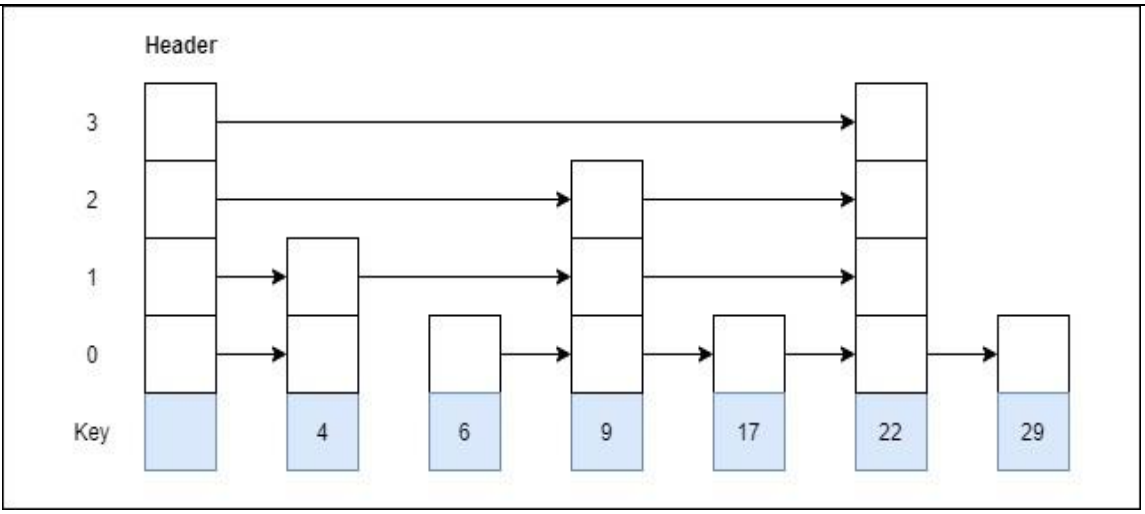
1. Write a program to implement Depth First Search
2. Write a program to implement Breadth First Search

Case Study: Select any five practical applications mentioned below

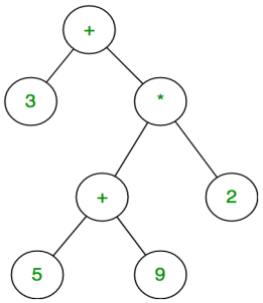
Note: A report has to be submitted by every student at the end of the semester that includes requirements, design, coding, and output with testing results of a real example.

1. Demonstrate to convert an infix expression into a postfix expression and evaluate to find the result.
2. Demonstrate to convert an infix expression into a prefix expression
3. Demonstrate a queue using Linked List and Stack.
4. Demonstrate a sparse matrix using array and linked list
5. Create a skip list, to insert these following keys in the empty skip list.
    a. 6 with level 1.
    b. 29 with level 1.
    c. 22 with level 4.
    d. 9 with level 3.
    e. 17 with level 1.
    f. 4 with level 2.

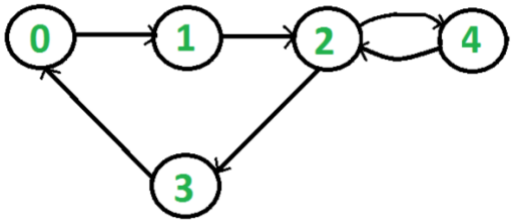
Implement all basic operations of skip list and demonstrate with examples. Skip list structure is shown below for reference.



6. Given an array representation of min Heap, convert it to max Heap and then apply Heapsort concept to display the data in decreasing order.
   Input: arr[ ] = {3, 5, 9, 6, 8, 20, 10, 12, 18, 9}
7. Make use of Radix sort algorithm to sort an array by individual digits, starting with the least significant digit.
8. Model a linked list data structure to add two polynomials.
9. Design a system to manage employee records {empID, empname, dept, salary}, and implement efficient basic operations based on employee ID.
10. Construct an expression tree i.e. a binary tree in which each internal node corresponds to the operator and each leaf node corresponds to the operand.
    For example expression tree for 3 + ((5+9)*2) would be:
    Demonstrate with required operations to convert this above expression into corresponding prefix, and postfix expressions and evaluate the result of the expression.



11. Demonstrate topological sorting for a Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge u-v, vertex u comes before v in the increasing order a vertex with no. of incoming edges.
12. Given a directed graph, check whether the graph contains a cycle or not. Your function should return true if the given graph contains at least one cycle, else return false. For example, the following graph contains two cycles 0->1->2->3->0 and 2->4->2. Demonstrate with required operations to display the results in the form of true and the cyclic path if any. Make use of BFA concept to solve this problem.



13. Find the frequency of each character in a string using Hashing Data Structure.

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-VII

Object Oriented Programming through Java Lab

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC15 **L-T-P-C: 0-0-3-1.5**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Apply fundamental Java programming constructs to develop basic applications. (L3)
CO2:	Construct object-oriented programs to develop modular and reusable software components. (L3)
CO3:	Develop robust applications by implementing exception handling, multithreading, techniques to enhance reliability and efficiency. (L3)
CO4:	Build interactive Graphical User Interfaces (GUIs), handle user events, and perform file operations to create user-friendly applications. (L3)
CO5:	Utilize Java Database Connectivity (JDBC) to manage data storage and retrieval in Java applications. (L3)

List of Experiments

Module-1: If Condition, Switch Case, Command Line Arguments

1. Write a Java program to determine whether a person is a Child, Teenager, or Adult based on their age. Children (age < 13) → Serve Water, Teenagers (13–19) → Serve Badam Milk, Adults (20 and above) → Serve Coke.
2. Write a Java program that reads two command-line arguments.
   If both are numbers, add them. If both are strings, concatenate them. If neither, print an error message.
3. Write a Java program that converts a student mark into words.
   Example: Input: 56 → Output: "Fifty-Six"(Assume valid marks are 0-100 only)

Module-2: Loops, Arrays

1. Write a Java program to print the first 50 ugly numbers.
2. Write a Java program to find leap years between 1947 and 2050.
3. Write a Java program to compute the difference between the largest and smallest array values.

Module-3: Strings

1. Write a Java program to reverse each word in a given sentence while maintaining the order of words.

2. Write a Java program to check if two given strings are anagrams. Two strings are anagrams if they contain the same characters in the same frequency, but in a different order (ignoring spaces and case sensitivity).

Module-4: Classes, Methods, Constructors

1. Write a Java program to illustrates Create a class Rectangle.The class has attributes length and width. It should have methods that calculate the perimeter and area of the rectangle. It should have readAttributes method to read length and width from user.
2. Write a Java program to create a class called Student with instance variables studentId, studentName, and grade. Implement a default constructor and a parameterized constructor that takes all three instance variables. Use constructor chaining to initialize the variables. Print the values of the variables.

Module-5: Inheritance, Polymorphism

1. Write a Java program to create a base class Vehicle with attributes like brand, model, and speed. Derive classes Car and Bike from Vehicle, each with unique attributes (e.g., Car has numDoors, and Bike has hasGear). Implement a method displayDetails() in each subclass to print specific details.
2. Write a Java program to create a base class BankAccount with methods deposit(), withdraw(), and calculateInterest(). Derive classes SavingsAccount and CurrentAccount. Override calculateInterest() in SavingsAccount to apply interest and in CurrentAccount to maintain a minimum balance condition.
3. Implement single and multi-level inheritance.
4. Demonstrate abstract classes with area calculation for shapes.

Module-6: Packages, Abstraction, Interface

1. Write a Java program to create an abstract class Medicine with a method displayLabel(). Derive classes Tablet, Syrup, and Ointment, each implementing displayLabel() with specific instructions (e.g., "Store in a cool place" for Tablet, "Shake well before use" for Syrup). Organize these classes in a package pharma.(Abstraction & Packages).
2. Write a Java program to define an interface BankAccount with methods deposit(), withdraw(), calculateInterest(), and displayBalance(). Implement this interface in SavingsAccount and CurrentAccount classes with specific logic. Place all classes in a package named banking.(Interface & Packages).
3. Write a Java program to create an abstract class Shape with an abstract method calculateArea(). Implement subclasses Circle, Rectangle, and Triangle to compute area based on user input. Use an interface Drawable with a method draw() to display the shape‘s type. Organize these classes in a package geometry.(Abstraction & Interface).

Module-7: Exception Handling, Multithreading

1. Write a Java program that describes exception handling mechanism with try, catch, finally, throw and throws.
2. Write a Java program to illustrate the concept of User defined Exceptions creation.
3. Write a Java program that creates threads by extending Thread class. First thread display ―Good Morning ―every 1 sec, the second thread displays ―Hello GVPW― every 2 seconds and the third display ―Welcome to JAVA‖ every 3 seconds, (Repeat the same by implementing Runnable).
4. Write a Java program to create a producer-consumer scenario using the wait() and notify() methods for thread synchronization.

Module-8: Event Handling & GUI Programming

1. Write a Java program to handle mouse and keyboard events.
2. Write a Java program to paint like paint brush in applet.
3. Write a Java program to demonstrate FlowLayout, BorderLayout, and GridLayout.

Module-9: File Handling

1. Write a Java program, which takes file/directory name from console and print whether it's a file or directory (if it exists). Also, print the details corresponding to it like is it readable,
2. writable, where on disk is it located (absolute path), etc).
3. Write a Java program to count lines, characters, and words from a given text file as input.

Module-10: JDBC

1. Write a Java program to connect to a MySQL database using JDBC. The program should include fields such as id, name, and email in a table named users.
2. Write a Java program to implement functionality to insert a new user into the database and retrieve all records from the user's table.

Case Study:

Select any one application mentioned below

Note: A report has to be submitted by every student at the end of the semester that includes design, coding, output, etc.

1. **Library Management System**

   Objective: Develop a system for managing books, borrowers, and transactions.

   OOP Concepts to be Applied:

   - **Encapsulation**: Book, Member, and Transaction classes with private attributes.
   - **Inheritance**: Subclasses like StudentMember and FacultyMember extending Member class.
   - **Polymorphism**: Overriding calculateFine() method for different types of members.
   - **Exception Handling:** Handling scenarios like issuing a book when none are

available.

- **JDBC**: Storing book and member details in a database.

Implementation Details:

- Book class with attributes like title, author, ISBN, and availability.
- Member class with attributes like memberID, name, contact.
- Transaction class to record book issues and returns.
- Library Management class to handle book issues, returns, and database interactions.

2. **E-Commerce Shopping Cart**

Objective: Implement a shopping cart system that allows customers to browse products, add to cart, and place orders.

OOP Concepts to be Applied:

- **Encapsulation:** Product and Order classes to store item details securely.
- **Inheritance:** ElectronicItem and ClothingItem subclasses extending Product.
- **Polymorphism:** Applying discount methods differently for different product categories.
- **Collections Framework**: Using ArrayList and HashMap for cart management.
- **JDBC:** Connecting the system with a database to store orders.

Implementation Details:

- Product class with productID, name, price, category.
- Cart class using HashMap<Product, Integer> for storing cart items.
- Order class to store order details.
- E-commerce System class to manage transactions and payment processing.

3. **Banking System**

Objective: Design a banking application to manage customer accounts, transactions, and loans.

OOP Concepts to be Applied:

- **Encapsulation:** Account details are private with public getter/setter methods.
- **Inheritance**: SavingsAccount, CurrentAccount, LoanAccount extending Account.
- **Polymorphism**: Overriding calculateInterest() for different accounts.
- **Exception Handling**: Handling insufficient balance, invalid transactions.
- **Multithreading**: Handling multiple transactions simultaneously.

Implementation Details:

- Account class with accountNo, balance, ownerName.
- Transaction class to manage deposits, withdrawals.
- Bank class with ArrayList<Account> to store customer accounts.

4. **Online Examination System Objective:** Build an online exam portal for students to take tests and view results.

OOP Concepts to be Applied:

- **Encapsulation**: Secure student and question data.
- **Inheritance**: MultipleChoiceQuestion and DescriptiveQuestion extending Question.
- **Polymorphism**: Implementing different grading strategies.
- **Event Handling**: Capturing user inputs for answers.
- **JDBC**: Storing student results in a database.

Implementation Details:

- Question class with questionID, questionText, options[], correctAnswer.
- Exam class to manage multiple questions.
- Student class with studentID, name, marks.

5. **Hotel Booking System**

Objective: Develop a system where users can book rooms, cancel bookings, and check availability.

OOP Concepts to be Applied:

- **Encapsulation**: Private room details with public methods.
- **Inheritance**: LuxuryRoom and EconomyRoom extending Room.
- **Polymorphism**: Overriding calculatePrice() for different room types.
- **JDBC**: Managing room bookings in a database.

Implementation Details:

- Room class with roomNumber, price, availability.
- Customer class storing customerID, name, contactDetails.
- Booking class handling room reservations.

Reference Books:

1. Herbert Schildt – Java: The Complete Reference (11th Edition), McGraw Hill, ISBN: 9 78-1260440232.
2. Cay S. Horstmann – Core Java Volume I – Fundamentals (8th Edition), Pearson, ISBN: 978-0135166307.
3. E. Balagurusamy – Object-Oriented Programming with Java (8th Edition), McGraw Hill, ISBN: 978-9353162344.
4. Paul Deitel & Harvey Deitel – Java: How to Program (11th Edition), Pearson, ISBN: 978-0134743356.

Web References:

1. https://docs.oracle.com/javase/tutorial/
2. https://onlinecourses.nptel.ac.in/noc20_cs58/preview

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-VIII

OPERATING SYSTEMS LAB

II Year B. Tech. I Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC17 **L-T-P-C: 0-0-3-1.5**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Make use of Linux commands for shell programming. (L3)
CO2:	Construct programs for various OS concepts and Linux commands using System Calls. (L3)
CO3:	Develop and implement programs to demonstrate various CPU Scheduling Algorithms, Process Synchronization Concepts in Operating Systems and Deadlock Prevention and Avoidance Techniques. (L3)
CO4:	Build programs for Memory Management Techniques and Page Replacement Algorithms. (L3)
CO5:	Develop programs for various DISK Scheduling Algorithms. (L3)

List of Experiments

Module-1: Implementation of basic Linux commands

1. Navigate to a directory, create a subdirectory, copy files into it, change permissions, and list them using various commands.
2. Create three empty text files using the touch command (e.g., file1.txt, file2.txt, file3.txt).
3. Use the cp command to copy one of the text files to another directory. If no other directory exists, create one using mkdir and then copy the file. Rename one of the text files using the mv command.
4. Create a text file in Vi that contains a list of files in your home directory. Include descriptions for each file. Use commands like :w, :q, :set number, and navigation commands to format it.
5. Modify the file using dd, yy, p, and /search commands. Save the file and quit.

Module-2: Write a C program on scheduling algorithms in Linux/Windows OS.

6. Implementation and Analysis of FCFS Scheduling Algorithm
7. Implementation and Analysis of SJF Scheduling Algorithm
8. Implementation and Analysis of Round Robin Scheduling Algorithm

Module-3: Implementation of Critical Section Problem Using Peterson's Algorithm

1. Mutual exclusion
2. Progress
3. Bounded waiting between two processes.

Module-4: Write simple shell programming in Linux OS.

1. Hello World Script
2. List Files in Directory
3. Check if a File Exists
4. Count Lines in a File
5. Sum of Numbers
6. Display Date and Time
7. Reverse a String
8. Calculate Factorial
9. Check Prime Number

Module-5: Shell programming using decision making constructs, loop constructs, file and directory manipulation

1. Decision-Making Constructs: Write a shell script that checks whether a number entered by the user is positive, negative, or zero. Use if, else, and elif statements for the decision making.
2. Loop Constructs: Write a shell script that prints the first 10 Fibonacci numbers using for loop.
3. File and Directory Manipulation: Write a shell script that accepts a file name from the user and checks whether it exists. If the file exists, display the number of lines in the file; otherwise, print "File does not exist."

Module-6: Simple C programs using command line arguments, system calls, library function calls, make utility in Linux OS.

1. **Command-Line Arguments: Simple Calculator**: **Description**
   Write a C program that accepts command-line arguments to perform basic arithmetic operations (addition, subtraction, multiplication, division). The user should provide the operation type (+, -, *, /) and two operands.
2. **System Calls: File Copy Using open(), read(), and write()**
   Write a C program that should copy a file from one location to another using system calls like open(), read(), and write().
3. **Library Function Calls: String Reversal**
   Write a C program that takes a string as input and returns the reversed string using standard library functions.

Module-7: Write C programs to study deadlock avoidance and detection in Linux/Windows OS.

Module-8: C programs using system call to create processes and study parent, child process mechanism in Linux OS.

1. Simple Parent-Child Process using fork().
2. Parent Waiting for Child to Exit using wait().

Module-9: Write a C Programs to simulate free space management in Linux/Windows OS.

1. first fit
2. best fit
3. worst fit

Module-10: Write a C programs to create process chaining, spawning in Linux OS.

1. Process Chaining using fork()
2. Process Spawning using fork() and exec()

Module-11: Write a C programs to error handling using errno(), perror() function in Linux OS.

1. File Opening Error
2. Division by Zero
3. Memory Allocation Error
4. Invalid File Descriptor

Module-12: Write C programs to use pipe system call for interprocess communication between Parent and Child Process in Linux OS**.**

Module-13: Write C programs for page replacement implementing in Linux/Windows OS.

1. FIFO
2. Optimal
3. LRU

Module-14: Write C programs for disk scheduling algorithms in Linux/Windows OS.

1. SCAN
2. SSTF
3. LOOK

Case Study:

Select any one case study mentioned below.

Note: At the end of the semester, each student is required to submit a report. Compare any of two operating systems such as Windows, Linux, macOS, Android, or others.

The report should include a case study covering the following topics:

1. Different operating systems implement their file systems in unique ways, depending on their architecture and design objectives. Write an overview of how file systems function across various operating systems.
2. Memory management is responsible for allocating and deallocating memory space to various programs and ensuring efficient utilization of system memory.
3. Different operating systems implement scheduling in unique ways based on their design goals and hardware requirements. Below is an overview of scheduling across various operating systems
4. Protection ensures the integrity, confidentiality, and availability of the system, preventing unintentional or intentional harm, an overview of how protection is implemented across different operating systems.
5. Each operating system handles I/O operations through a combination of system calls, device drivers, and I/O subsystems to provide seamless interaction with hardware, an overview of how different operating systems manage I/O operations input/output.

Reference Books:

1. Unix concepts and applications by Sumitabha Das, TMH Publications.
2. Unix programming by Stevens, Pearson Education.
3. Shell programming by Yashwanth Kanetkar.
4. Operating System Concepts by Silberschatz, and Peter Galvin.

Web References:

1. https://nptel.ac.in/courses/106106144
2. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-828- operating- system-engineering-fall-2012/
3. https://cs140.stanford.edu

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-IX

ADVANCED PYTHON PROGRAMMING LAB

II Year B. Tech. I Semester

[Skill Course]

Course Code: 24CS11SC01 **L-T-P-C: 1-0-2-2**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Demonstrate the ability to create, manipulate, and analyze NumPy arrays and matrices for efficient data handling and numerical computations. (L2)
CO2:	Develop Pandas Series and DataFrames to analyze and process structured data, including string operations. (L3)
CO3:	Apply data integration techniques to merge, deduplicate, and organize datasets for analysis. (L3)
CO4:	Make use of Matplotlib to identify patterns, trends, and insights in structured datasets and visualize them. (L3)
CO5:	Build interactive applications that combine user inputs, data processing, and visualizations in Streamlit. (L3)

List of Experiments

Module-1: Numpy Library- Install Numpy and perform the following tasks

1. Create a numpy array from list, tuple with float type
2. Python program to demonstrate slicing, integer and boolean array indexing
3. Write a python program to find min, max, sum, cumulative sum of array.
4. Write a python program to demonstrate use of ndim, shape, size, dtype.

Module-2: Numpy Library- Linear Algebra

1. Write a python program to find rank, determinant, and trace of an array.
2. Write a python program to find eigenvalues of matrices
3. Write a python program to find matrix and vector products (dot, inner, outer, product), matrix exponentiation.

4. Write a python program to solve a linear matrix equation, or system of linear scalar equations.

Module-3: Pandas DataSeries and DataFrame- Install pandas and perform the following tasks

1. Write a python program to implement Pandas Series with labels.
2. Create a Pandas Series from a dictionary.
3. Creating a Pandas DataFrame.
4. Write a program which make use of following Pandas methods
   i)describe(), ii) head() iii) tail()

Module-4: Pandas String and Regular Expressions

1. Write a Pandas program to convert all the string values to upper, lower cases in a given pandas series. Also find the length of the string values.
2. Write a Pandas program to remove whitespaces, left sided whitespaces and right sided whitespaces of the string values of a given pandas series.
3. Write a Pandas program to count of occurrence of a specified substring in a DataFrame column.
4. Write a Pandas program to swap the cases of a specified character column in a given DataFrame.

Module-5: Data Wrangling operations

1. Consider a DataFrame with multiple columns and perform the following tasks:
2. Identify columns with missing values, count the total missing values in each. Then, fill missing values in numerical columns with the median of each respective column.
3. Perform an outer merge of two DataFrames on a common column to include all rows from both DataFrames. After merging, identify and handle any duplicated rows that arise from the merge process.
4. Apply boolean indexing to filter rows based on multiple conditions (e.g., a column value greater than a threshold and a string column contains a specific substring.).
5. Use. loc[] to retrieve rows and columns based on specific conditions, and .iloc[] to retrieve rows and columns by integer position.

Module-6: Pandas Grouping Aggregate

Consider a dataset containing columns such as Category, ID, Value1, and Value2. Perform the following tasks:

1. Group the data by a categorical column.
2. Check and print the type of the groupby object before applying any aggregation function.

3. For each group, calculate: value of a numerical column.
4. value of a numerical column of another numerical column.
5. numerical column, ming the aggregation, check and print the type of the resulting object.

Module-7: Pandas Styling &Excel

1. Create a DataFrame of ten rows, four columns with random values. Write a Pandas program to highlight the negative numbers with red and positive numbers with black colours.
2. Create a DataFrame of ten rows, four columns with random values. Write a Pandas program to highlight the maximum value in each column.
3. Write a Pandas program to import excel data into a Pandas DataFram
4. Write a Pandas program to find the sum, mean, max, min value of a column of file.

Module-8: Visualization using Matplotlib

Given a DataFrame with Product and Sales_Amount columns, create a bar plot to visualize the total sales for each product. Label the x-axis as "Product" and the y-axis as "Total Sales".

1. Given a DataFrame with an Age column, create a histogram to visualize the distribution of ages using 10 bins. Label the x-axis as "Age Range" and the y-axis as "Frequency". Add a legend to indicate the number of bins used and annotate the plot with the mean age value.
2. Create a line plot for Temperature over Date and annotate the highest and lowest temperatures.
3. Create a scatter plot to show the relationship between Height and Weight in a DataFrame. Use different colors for different age groups (e.g., "Under 20", "20-40", "40+"), and add a legend to indicate the age group categories. Label the axes as "Height" and "Weight".

Module-9: Introduction to Streamlit: install streamlit library and perform the following

1. Write a Streamlit app that displays a title, a header, and some text using st.title(), st.header(), and st.write().
2. Create a Streamlit app where the user enters their name using st.text_input(), and the app displays a greeting message (e.g., "Hello, [name]!").
3. Develop a Streamlit app with a button (st.button()). When clicked, it should display a message like "Button clicked!".
4. Write a Streamlit app that loads and displays an image from a file using st.image().

Module-10: Streamlit Widgets & Data Handling

1. Create a Streamlit app where a user selects a number using st.slider(), and the app displays the square of that number.

2. Develop a Streamlit app with a checkbox (st.checkbox()). If checked, display a message like "Checkbox is checked!".
3. Build a Streamlit app that allows users to upload a CSV file using st.file_uploader(), and display the first five rows using Pandas.
4. Write a Streamlit app that generates a random dataset using NumPy and plots a simple line chart using Matplotlib and st.pyplot().

Project Instructions: Data Analysis with NumPy and Pandas

Note: A report has to be submitted by every student at the end of the semester that includes the following:

Objective:

Develop a **Streamlit** application by taking datasets and analyze them using **NumPy** and **Pandas**. Perform various data wrangling operations, analysis, and visualization tasks to derive meaningful insights from the data.

Dataset:

Choose a dataset from sources like Kaggle, UCI Machine Learning Repository, or any other dataset available online. Examples include:

- Sales data
- Financial data
- Weather data
- Customer or e-commerce data
- Health or medical datasets

Tasks:

1. **Data Loading & Inspection:** Load the dataset into Pandas. Inspect data using .head(), .info(), .describe(). Check for missing values using NumPy.
2. **Data Cleaning:** Handle missing values (drop, fill). Remove duplicate entries. Ensure data consistency.
3. **Data Wrangling:** Convert data types if necessary. Create new derived columns. Apply transformations like normalization or encoding.
4. **Filtering & Indexing:** Select relevant data using Pandas and NumPy. Apply logical conditions for filtering. Set meaningful indexes.
5. **Aggregation & Grouping:** Group data by relevant categories. Compute aggregations like sum, mean, count, etc.
6. **Visualization:** Use Matplotlib for plots (line, scatter, bar, histogram). Add annotations and legends. Ensure visualizations effectively communicate insights.
7. **Develop Streamlit App:** Build an interactive web app using Streamlit. Allow users to upload datasets and explore insights. Provide interactive filtering, grouping, and visualization.
8. **Final Report:** Summarize key insights with clear explanations. Include visualizations to support findings. Explain important data wrangling steps.

Reference Books:
1. Martin C. Brown (Author), —Python: The Complete Reference‖ McGraw Hill Education, Fourth edition , 2018 2. R. Nageswara Rao , —Core Python Programming‖ Dreamtech Press India Pvt Ltd 2018.
Web References:
1. https://onlinecourses.nptel.ac.in/noc19_cs40/preview 2. https://onlinecourses.nptel.ac.in/noc19_cs41/preview

COURSE STRUCTURE

(Applicable for the Academic Year 2024-25 onwards)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.Tech. Computer Science and Engineering
SCHEME AND SYLLABI
(With effect from 2024-25 admitted batch)

II Year - II Semester

Course Code	Category	Course Title	L	P	Internal Marks	External Marks	Total Marks	Credits
24BM11RC06	BS	Probability and Statistics	3	0	30	70	100	3
24HM11RC01	HSS	Managerial Economics	3	0	30	70	100	3
24CT11RC18	PC	Design and Analysis of Algorithms	3	0	30	70	100	3
24CT11RC12	PC	Database Management Systems	3	0	30	70	100	3
24CT11RC20	PC	Formal Languages and Automata Theory	3	0	30	70	100	3
24CT11RC21	PC	Algorithms Lab Through C++	0	3	50	50	100	1.5
24CT11RC16	PC	Database Management Systems Lab	0	3	50	50	100	1.5
24CT11SC02	SC	Skill Course II: MERN Stack Development	1	2	50	50	100	2

24HM11MC01	MC	Professional Ethics and Universal Human Values	2	0	50	50	100	**0**
24CS11SW01	MC	NCC/NSS	2	0	50	50	100	**0**
							Total Credits	**20**

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-X

DESIGN AND ANALYSIS OF ALGORITHMS

II Year B. Tech. II Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC18 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Analyse the time and space complexities of algorithms using growth of functions. (L4)
CO2:	Apply and analyse divide and conquer and greedy paradigm to generate efficient solutions of real-time problems. (L3, L4)
CO3:	Develop and analyse algorithms using dynamic programming paradigm to generate optimal solutions of real-time applications. (L3, L4)
CO4:	Develop and analyse algorithms using Backtracking paradigm to generate optimal solutions of real-time applications. (L3, L4)
CO5:	Develop and analyse algorithms using Branch and Bound paradigm to generate optimal solutions of real-time applications. Classify polynomial and non-polynomial problems. (L3, L4)

UNIT-I	**8 Lectures**
Introduction: Algorithm Criteria**,** Algorithm Specification, Performance Analysis-Space Complexity, Time Complexity, Asymptotic Notations, Amortized Analysis.	

UNIT-II	**12 Lectures**
Divide and Conquer: General Method, The master method for solving recurrences, Binary Search, Merge Sort, Quick Sort, Strassen's Matrix Multiplication. **The Greedy Method**: The General Method, Knapsack Problem, Job Sequencing with Deadlines, Minimum-Cost Spanning Trees, Single Source Shortest Paths.	

UNIT-III	**10 Lectures**
Dynamic Programming: The General Method, Multistage Graphs, All Pairs-Shortest Paths, Optimal Binary Search Trees, 0/1 Knapsack Problem, Reliability Design, The Traveling Salesperson Problem.	

UNIT-IV	**10 Lectures**
Backtracking: The General Method, The 8-Queens Problem, Sum of Subsets, Graph Coloring, Hamiltonian Cycles, Knapsack Problem.	

UNIT-V	**10 Lectures**
Branch and Bound: The Method - Least Cost (LC) Search, The 15-puzzle, Control Abstraction for LC- Search, Bounding, FIFO Branch-and-Bound, LC Branch-and-Bound, 0/1 Knapsack Problem, Traveling Salesperson Problem. **NP-Hard and NP-Complete problems**: Basic concepts, Non-Deterministic Algorithms, The Classes NP-hard and NP-Complete, Cook's Theorem.	

Text Books:
1. Introduction to Algorithms, 3rd Edition edition by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, PHI. 2. Fundamentals of Computer Algorithms‖, 2nd Edition, Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, Universities Press.
Reference Books:
1. Mark Allen Weiss, Data Structures and Algorithm Analysis in C++, Pearson, 4th edition 2. S. Sridhar, —Design and Analysis of Algorithms‖, Oxford University Press.
Web References:
1. http://nptel.ac.in/courses/106101060/

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-XI

DATABASE MANAGEMENT SYSTEMS

II Year B. Tech. II Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC12 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Explain the basic concepts of Database Management System, levels of abstraction, architecture of DBMS. (L2)
CO2:	Construct the E-R model to perform conceptual design and summarize the concepts of Basic SQL, Relational Algebra, Relational Calculus to manipulate the database. (L3)
CO3:	Apply Relational Model concepts to perform various operations. (L3)
CO4:	Relate and Plan the concept of data planning and database design using normalization. (L2)
CO5:	Make use of the ACID properties, concurrency control mechanisms, File Organizations and Indexing Techniques. (L3)

UNIT-I	**10 Lectures**
Introduction: Database system, Characteristics (Database Vs File System), Advantages of Database systems, Database Applications. Describing and Storing Data in a DBMS - Relational Model, Levels of abstraction, Data Independence, Structure of a DBMS, People Who Work with Databases.	

UNIT-II	**10 Lectures**
Introduction to Database Design: Database Design and E-R Diagrams, Entities, Attributes and Entity Sets, Relationship and Relationship Sets, Conceptual Design With the E-R Models. **Basic SQL:** Simple Database schema, data types, table definitions (create, alter), Creating tables with relationship, Implementation of key and integrity constraints different DML operations (insert, delete, update), basic SQL querying (select and project) using where clause, arithmetic & logical operations, SQL functions (Date and Time, Numeric, String conversion). **Relational Algebra and Calculus**: Relational Algebra-Selection and Projection, Set Operation, Renaming, Joins and Division, Relational Calculus.	

UNIT-III	**12 Lectures**
Relational Model: Introduction to relational model, concepts of domain, attribute, tuple, relation, importance of null values, The Relational Model Integrity Constraints Over Relations- Key Constraints–Foreign Key Constraints-General Constraints, nested queries, sub queries, grouping, aggregation, ordering, implementation of different types of joins, view (updatable and non updatable), relational set operations, Complex Integrity Constraints in SQL, Triggers and Active Databases.	

UNIT-IV	**10 Lectures**
Schema Refinement and Normal Forms: Introduction to schema refinement, functional dependency, Reasoning abouts FDs, Normal forms-1NF, 2NF, 3NF and Boyce-codd normal form (BCNF), Properties of Decomposition- Lossless join and dependency preserving decomposition, Fourth normal form(4NF), Fifth Normal Form (5NF).	

UNIT-V	**10 Lectures**
Transaction Management and Concurrency Control: Transaction, ACID properties, transaction log and transaction management with SQL using commit rollback and save point, Transactions and Schedules, Concurrent Execution of Transactions, Concurrency control with locking methods-lock granularity, lock types, two phase locking for ensuring serializability, deadlocks, Concurrency control with time stamp ordering- Wait/Die and Wound/Wait Schemes Overview of Storage and Indexing - Data on External Storage, File Organizations and Indexing, Index data Structures.	

Text Books:

1. Database Management Systems, 3/e, Raghurama Krishnan, Johannes Gehrke, TMH.
2. Database System Concepts,5/e, Silberschatz, Korth, TMH.

Reference Books:

1. Introduction to Database Systems, 8/e C J Date, PEA.
2. Database Management System, 6/e Ramez Elmasri, Shamkant B. Navathe, PEA.
3. Database Principles Fundamentals of Design Implementation and Management, Corlos Coronel, Steven Morris, Peter Robb, Cengage Learning.

Web References:

1. https://nptel.ac.in/courses/106/105/106105175/

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-XII

FORMAL LANGUAGES AND AUTOMATA THEORY

II Year B. Tech. II Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC20 **L-T-P-C: 3-0-0-3**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	**Construct** a Deterministic Finite Automata (DFA) and a Non-Deterministic Finite Automata (NFA) for a given regular language and compare their structures. (L3)
CO2:	**Make Use** of regular expressions to generate finite automata and demonstrate how to apply algebraic laws to simplify complex regular expressions. (L3)
CO3:	**Construct** parse trees for context-free grammars and identify ambiguities in grammars and their corresponding languages. (L3)
CO4:	**Identify** the language recognized by a Pushdown Automaton by analyzing its transitions and stack operations. (L3)
CO5:	**Solve** computational problems by applying Turing machine constructions to model the problem-solving process. (L3)

UNIT-I	**12 Lectures**
Finite Automata: Need for Automata Theory , The Central Concepts of Automata Theory, Grammar, Types of Grammars, Chomsky Hierarchy, Finite Automata -An Informal Picture of Finite Automata, Deterministic Finite Automata(DFA), Non-Deterministic Finite Automata (NFA), Design of NFA, Equivalence of DFA and NFA, Finite Automata with €- Transitions**,** Minimization of Finite Automata, Mealy and Moore Machines, Applications and Limitation of Finite Automata.	

UNIT-II	**8 Lectures**
Regular Expressions: Regular Expressions**,** Regular Sets, Identity Rules, Algebraic Laws for Regular Expressions**,** Finite Automata and Regular Expressions**,** Pumping Lemma**,** Closure Properties, Regular Grammar-Equivalence of Finite Automata and Regular Grammars, Converting Finite Automation to Regular Grammar.	

UNIT-III	**10 Lectures**
Context Free Grammars and Languages: Context Free Grammars-Definition of Context Free Grammar, Derivations using a Grammar, Leftmost and Rightmost Derivations, The Language of a Grammar, Sequential Forms, Parse Trees, Ambiguity in Grammars and Languages-Ambiguous Grammars, Removing Ambiguity from Grammars. **Simplification of Context Free Grammars**-Elimination of Useless Symbols, €- productions and Unit Productions, Normal Forms for Context Free Grammars-Chomsky Normal Form and Greibach Normal Form, Pumping Lemma**,** Closure Properties, Applications of Context Free Grammars.	

UNIT-IV	**8 Lectures**
Pushdown Automata: Definition of Pushdown Automata-The Formal Definition of PDA, Graphical Notation of PDA, Instantaneous Description of Push Down Automata, The Languages of PDA, Equivalence of Pushdown Automata and Context Free Grammars Conversion-From Grammar to Pushdown Automata, From PDA to Grammar, Deterministic Pushdown Automata.	

UNIT-V	**12 Lectures**
Turing Machine: Turing Machine as a Computational Machine, Techniques for Turing Machine Construction, Types of Turing Machines**,** Universal Turing Machine, Recursive and Recursively Enumerable Languages. **Undecidability and Computability**: Decision Problems, Decidability and Decidable Languages, Halting Problem, Post Correspondence Problem**.** **Non-Deterministic Polynomial Completeness:** P-Problems, NP-Problems, NP- Hard and NP-Complete Problems**.**	

Text Books:

1. Introduction to Automata Theory, Languages and Computation, J.E.Hopcroft, R.Motwani and J.D.Ullman, 3rd Edition, Pearson, 2008.
2. Formal Language and Automata Theory, K.V.N.Sunitha and N.Kalyani, Pearson, 2015.
3. Theory of Computer Science-Automata, Languages and Computation, K.L.P.Mishra and N.Chandrasekharan, 3rd Edition, PHI, 2007.

Reference Books:

1. Introduction to Automata Theory, Formal Languages and Computation, Shyamalendu Kandar, Pearson, 2013.
2. Introduction to the Theory of Computation, Michael Sipser, 3rd Edition, Cengage earning, 2012.

Web References:

1. https://archive.nptel.ac.in/courses/106/105/106105196/
2. https://archive.nptel.ac.in/courses/106/106/106106049/
3. https://www.coursera.org/courses?query=theory+of+computation
4. https://www.classcentral.com/course/swayam-formal-language-and-automata- theory-an-application-in-compiler-design-291835
5. https://infolab.stanford.edu/~ullman/ialc.html

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-XIII

ALGORITHMS LAB THROUGH C++
II Year B. Tech. II Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC21 **L-T-P-C: 0-0-3-1.5**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Apply the object-oriented concepts such as class, object, inheritance, encapsulation and other features. (L3)
CO2:	Develop the code to demonstrate merge sort, quick sort and other applications using divide and conquer technique. (L3)
CO3:	Make use of Greedy and Dynamic Programming techniques to design and implement various optimal problems. (L3)
CO4:	Utilize backtracking technique to solve the optimal problems like sum of subsets and Hamiltonian cycles. (L3)
CO5:	Build the optimal problems like Travelling Sales Person Problem using branch and bound technique. (L3)

List of Experiments

Module-1: Basics of C++

1. Write a C++ program to demonstrate the keywords continue and break.
2. Read a value n=10 and display 1, 2, 3, 5, 6, 8, 10 and 1, 2, 3, 4, 5
3. Write a C++ program to create a Calculator to find addition, subtraction, multiplication, division, or remainder of two given numbers using the switch Statement as user‘s choice.
4. Write a C++ program to demonstrate call-by-value and call-by reference concepts to swap two numbers.
5. Write a C++ program to demonstrate function overloading by using function area with different combination of required parameters to calculate the area of a square, rectangle, and circle by taking suitable input values as float or integer.

Module-2: OOP concepts using C++

1. Write a C++ program using a class called Student with the following attributes within it.
   student_roll, student_name, programme_name, phone_number, grade
   Create n=7 Student objects and print the student_roll, student_name, programme_name, phone_number, and grade>8.0 of these objects with suitable headings. Demonstrate the program using constructor and destructor.

2. Write a C++ program to read two integers a and b. Compute a/b and print, when b is not zero. Raise an exception when b is equal to zero.

Module-3: Inheritance using C++

1. Write a C++ program to design a superclass called Staff with details as StaffId, Name, Phone, and Salary. Extend this class by writing three subclasses namely Teaching (branch, highest_degree, no_of_publications), Technical (skills), and Contract (period). Read and display at least 3 staff objects of all three categories.

   Demonstrate above program by deriving subclasses using all three access modes (public, private, and protected).

Module-4: Divide-and-Conquer

1. Write a C++ program to demonstrate following sorting techniques
   a. Merge Sort
   b. Quick Sort

   Read at least 15 numbers in a range of 1 to 100 and display the sorted output.
2. Write a C++ program to find the Maximum and Minimum element in the given Array

Module-5: The Greedy Method

1. Write a C++ program to find solution for job sequencing with deadlines problem.
2. Write a C++ program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.
3. Write a C++ program to perform Single source shortest path problem for a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

Module-6: Dynamic Programming

1. Write a C++ program to implement following algorithms using Dynamic Programming Concept.
   a. All-Pairs Shortest Paths problem using Floyd's algorithm
   b. 0/1 Knapsack problem

Module-7: Backtracking:

1. Write a C++ program to implement following algorithms using Backtracking Concept.
   a. Sum of subsets problem for a given set of distinct numbers
   b. Hamiltonian cycle problem.

Module-8: Branch and Bound:

1. Write a C++ program to implement following algorithm using Branch and Bound Concept.
   a. Travelling Sales Person problem.

Case Study:

Select any five practical applications mentioned below

Note: A report has to be submitted by every student at the end of the semester that includes design, coding, output, etc.

1. To Sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n>= 5000 and record the time taken to sort. Plot a graph of the time taken versus non graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using CPP how the divide and conquer method works along with its time complexity analysis: worst case, average case and best case.
2. To sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n>= 5000, and record the time taken to sort. Plot a graph of the time taken versus non graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using CPP how the divide and conquer method works along with its time complexity analysis: worst case, average case and best case.
3. Write a C++ program to implement Strassen‘s Matrix Multiplication Algorithm using the divide and conquer technique.
4. Write a C++ program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal‘s algorithm.
5. Write a C++ program to implement reliability optimization in system design using Dynamic Programming (DP). The goal is to select a combination of components that provides the maximum system reliability while staying within a given budget.
6. Write a C++ program to implement the Optimal Binary Search Tree (OBST) algorithm using Dynamic Programming (DP). The goal is to arrange search terms based on their frequencies so that frequent queries are retrieved faster, optimizing the auto-complete feature in a search engine.
7. Write a C++ program to implement the Graph Coloring Problem using the Backtracking Algorithm for scheduling university exams. The goal is to assign the minimum number of time slots (colors) to subjects such that no two exams with common students (conflicting subjects) are scheduled at the same time.
8. Write a C++ program to implement the N-Queens Problem using the Backtracking Algorithm to arrange students in a classroom, exam hall, or conference room while

maintaining social distancing. The goal is to ensure that: No two students sit in the same row, No two students sit in the same column, No two students sit diagonally adjacent.

9. Write a C++ program to solve the 0/1 Knapsack Problem using the Branch and Bound approach. Consider a scenario where a relief truck has a fixed weight capacity, and you must select essential supplies (food, medicine, blankets, etc.) to maximize the total priority value while staying within the weight limit.
10. Write a C++ program to solve the 15-Puzzle Problem using the Branch and Bound approach. The goal is to arrange the tiles in ascending order (from 1 to 15) with the blank space (0) at the bottom-right corner by sliding tiles into the empty space using the fewest moves.

Reference Books:

1. T. H. Cormen, C. L. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, MIT Press.
2. J. Kleinberg and E. Tardos, Algorithm Design, Addison-Wesley.
3. Harry R. Lewis and Larry Denenberg, Data Structures and Their Algorithms, Harper Collins.
4. A. Gibbons, Algorithmic Graph Theory, Cambridge University Press.
5. Michael T. Goodrich and Roberto Tamassia, Algorithm Design: Foundations, Analysis, and Internet Examples, John Wiley.
6. R. Sedgewick, Algorithms in C (Parts 1-5), Addison Wesley.
7. M. H. Alsuwaiyel, Algorithm Design Techniques and Analysis, World Scientific.
8. Gilles Brassard and Paul Bratley, Algorithmics: theory and practice, Prentice-Hall.
9. Udi Manber, Introduction to Algorithms: A Creative Approach, Addison-Wesley.
10. Sara Baase and Allen Van Gelder, Computer Algorithms: Introduction to Design and Analysis, Addison-Wesley.

Web References:

1. https://archive.nptel.ac.in/courses/106/105/106105164
2. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to- algorithms-fall-2011/

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
(Autonomous)
(Approved by AICTE, New Delhi and Permanently Affiliated to Andhra University, Visakhapatnam)
Madhurawada :: Visakhapatnam – 530 048

ANNEXURE-XIV

DATABASE MANAGEMENT SYSTEMS LAB

II Year B. Tech. II Semester

[Common to CSE, CSE (AI&ML), IT]

Course Code: 24CT11RC16 **L-T-P-C: 0-0-3-1.5**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Build the database using SQL DDL, DML commands and make use of built-in-functions to write queries. (L3)
CO2:	Identify and implement different operators and Apply nested queries to solve real time problems. (L3)
CO3:	Plan Queries on Joins, Aggregate functions, views. (L3)
CO4:	Construct simple PL/SQL programs using control statements and exception handling methods. (L3)
CO5:	Develop applications using PL/SQL including procedures, functions, cursors, packages and triggers. (L3)

List of Experiments

Implement the following modules using ORACLE

Module-1:

Creation, altering and dropping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.

Module-2:

Queries using Conversion functions (to_char, to_number and to_date), string functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), date functions (Sysdate, next_day, add_months, last_day, months_between, least, greatest, trunc, round, to_char, to_date)

Module-3:

Queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSECT Constraints

Module-4:

Queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, ORDER BY, HAVING clauses.

Module-5: Queries on Views and Joins

Module-6: PL/SQL

1. Create a simple PL/SQL program which includes declaration section, executable section and exception Handling section (Ex. Student marks can be selected from the table and printed for those who secured first class and an exception can be raised if no records were found)
2. Insert data into student table and use COMMIT, ROLLBACK and SAVEPOINT in PL/SQL block.

Module-7:

Develop a program that includes the features NESTED IF, CASE and CASE expression. The program can be extended using the NULLIF and COALESCE functions.

Module-8:

Program development using WHILE LOOPS, numeric FOR LOOPS, nested loops using ERROR Handling, BUILT–IN Exceptions, and USER DEFINED Exceptions, RAISE_APPLICATION _ERROR

Module-9:

Programs development using creation of procedures, passing parameters IN and OUT of Procedures

Module-10:

Program development using creation of functions, invoke functions in SQL Statements and write complex functions

Module-11:

1. Develop programs using features parameters in a CURSOR, FOR UPDATE CURSOR, WHERE CURRENT of clause and CURSOR variables.

2. Develop Programs using BEFORE and AFTER Triggers, Row and Statement Triggers and INSTEAD OF Triggers.

Case Study:

Select any one application mentioned below and Apply the Database Design steps

Note: A report has to be submitted by every student at the end of the semester

1. Accounting Package for Shops
2. Database Manager for Magazine Agency or Newspaper Agency
3. Ticket Booking for Performances
4. Preparing Greeting Cards & Birthday Cards
5. Personal Accounts - Insurance, Loans, Mortgage Payments, Etc.,

6. Doctor's Diary & Billing System
7. Personal Bank Account
8. Class Marks Management
9. Hostel Accounting
10. Video Tape Library
11. History of Cricket Scores
12. Cable TV Transmission Program Manager
13. Personal Library
14. Sailors Database
15. Suppliers and Parts Database

Reference Books:

1. Oracle: The Complete Reference by Oracle Press.
2. Nilesh Shah, "Database Systems Using Oracle‖, PHI, 2007.
3. Rick F Vander Lans, ―Introduction to SQL‖, Fourth Edition, Pearson Education, 2007.

Web References:

1. https://docs.oracle.com/en/
2. https://www.w3schools.com/sql/sql_exercises.asp
3. https://www.w3resource.com/sql-exercises/

ANNEXURE-XV

MERN STACK DEVELOPMENT

II Year B. Tech. II semester

[Skill Course]

Course Code: 24CT11SC02 **L-T-P-C: 1-0-2-2**

Course Outcomes: At the end of the Course, the student shall be able to	
CO1:	Illustrate the fundamental concepts of JavaScript, TypeScript, and asynchronous programming with examples. (L2)
CO2:	Identify the components of MERN architecture, including MongoDB, Express.js, React.js, and Node.js, and their roles in web application development. (L3)
CO3:	Demonstrate CRUD operations using MongoDB and Mongoose for database management. (L2)
CO4:	Build RESTful APIs using Node.js and Express.js to enable server-side communication. (L3)
CO5:	Develop a full-stack web application with authentication and state management, integrating React.js frontend with a Node.js backend. (L3)

List of Experiments

Module -1: JavaScript Fundamentals

1. **Javascript Fundamentals:** Difference between Javascript and ECMAScript, Constants, Variables, Arrays, Objects, Functions, Arrow Functions, Modules, Spreading operators and destructiong assignment.
2. **Typescript:** Adding Typescript support to ECMAScript, Types, Interfaces

3. **Asynchronous Programming:** Callbacks, Promises, Async/Await

Module-2: Introduction to MERN

1. Introduction to 3-Tier client/server architecture
2. Introduction to NoSQL
3. Introduction to REST API
4. Introduction to Web UI
5. Introduction to JSON
6. Overview of MERN Stac(MongoDB, Express.js, React, Node.js)

Module-3: MongoDB - NoSQL Database

1. Introduction to MongoDB and its features
2. Setting up MongoDB locally and on the cloud (MongoDB Atlas) Designing Collections and Documents
3. CRUD operations using MongoDB
4. Mongoose ODM for MongoDB schema and validation Aggregation framework and indexing

Module-4: Node.js - Server-Side Development

1. Setting up Node.js
2. Understanding Package Managers (npm, yarn)
3. Understanding package.json
4. Understanding Node.js
5. Architecture Create a Web Server in Node.js Restarting
6. Node Application
7. File System & Module Management

Module-5: Express.js – Backend Framework

1. Express Development Environment Defining and Handling Routes Route and Query Parameters.
2. Middleware (Custom, Built-in, Third-Party)
3. Connecting to MongoDB with Mongoose CRUD Operations
4. REST API Development
5. Testing API with Postman

Module-6: Angular.js – Frontend Development

1. Introduction to Angular.js & JSX
2. Environment Setup- installing Angular CLI
3. Create your first Angular app and understand its project structure
4. Work with components, directives, and pipes

Module-7: React.js – Frontend Development

1. Introduction to React.js & JSX
2. Functional Components
3. Nested Components
4. Properties and Property Drilling
5. Context Management with useContext
6. State Management using useState and useEffect
7. Global State Management with useReducer
8. Routing with React Router
9. Styling with Tailwind

Module-8: Enterprise Application Development

1. Connecting the UI with Backend
2. Client requests using Fetch API
3. Client requests using Axios
4. Security and JWT
5. Preparing WebUI for distribution Deployment strategies

Case Study:

Select any one practical application mentioned below

Note: A report has to be submitted by every student at the end of the semester that includes design, coding, output, etc.

Building a Real-World MERN Stack Application with Authentication & CRUD Operations for the following problem statements Tech Stack:

1. **Frontend**: React.js, React Router, Axios
2. **Backend:** Node.js, Express.js, MongoDB, Mongoose.
3. **Authentication:** JWT (JSON Web Tokens), bcrypt (for password hashing).
4. **Database:** MongoDB (for storing users and tasks).

1. Healthcare Appointment Scheduling System:
A hospital faced challenges in managing patient appointments, leading to long wait times and inefficient resource utilization.

2. Food Delivery Service Expansion:
A food delivery startup struggled to scale operations to new cities due to logistical challenges.

3. Smart Agriculture System:
Farmers faced challenges in monitoring crop health and optimizing water usage.

4. Online Learning Platform for Kids:
Parents struggled to find engaging and educational content for their children during the pandemic.

5. Waste Management App:
A city faced challenges in managing waste collection and recycling efforts.

Text Books:

1. Full Stack JavaScript Development with MEAN, Colin J Ihrig, Adam Bretz, 1st edition, SitePoint, SitePoint Pty. Ltd., O'Reilly Media.
2. MongoDB – The Definitive Guide, 2nd Edition, Kristina Chodorow,O'Reilly.

Reference Books:

1. Robert W. Sebesta, ―Programming with World Wide Web‖, Fourth Edition, Pearson, 2008.
2. Dayley B., ―Node.js, MongoDB, and AngularJS Web Development‖, Addison Wesley Professional, 2014.
3. Vainikka J., ―Full-Stack Web Development using Django REST Framework and React‖, 2018 .

Web References:

1. https://infyspringboard.onwingspan.com/en/app/toc/lex_17739732834840810000_shared/ove rview(HTML5)
2. https://infyspringboard.onwingspan.com/en/app/toc/lex_18109698366332810000_shared/ove rview(Javascript)
3. https://infyspringboard.onwingspan.com/en/app/toc/lex_32407835671946760000_shared/ove rview(Node.js & Express.js)
4. https://infyspringboard.onwingspan.com/en/app/toc/lex_9436233116512678000_shared/over view(Typescript)
5. https://infyspringboard.onwingspan.com/en/app/toc/lex_20858515543254600000_shared_/ overview (Angular JS)
6. https://infyspringboard.onwingspan.com/en/app/toc/lex_auth_013177169294712832113_shared/overview (MongoDB)