|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | Venkataramana Veeramsetty | | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | Dr. V. Venkataramana (Co-ordinator) | | Dr. T. Sampath Kumar | | Dr. Pramoda Patro | | Dr. Brij Kishor Tiwari | | Dr.J.Ravichander | | Dr. Mohammand Ali Shaik | | Dr. Anirodh Kumar | | Mr. S.Naresh Kumar | | Dr. RAJESH VELPULA | | Mr. Kundhan Kumar | | Ms. Ch.Rajitha | | Mr. M Prakash | | Mr. B.Raju | | Intern 1 (Dharma teja) | | Intern 2 (Sai Prasad) | | Intern 3 (Sowmya) | | NS\_2 ( Mounika) | | | | | | |
| **Course Code** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week6 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **AssignmentNumber:12.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
| **Name: Gundu Meghana**  **Roll NO: 2403A510C1**  **Batch: 04** | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 12: Algorithms with AI Assistance – Sorting, Searching, and Optimizing Algorithms**  **Lab Objectives:**   * Apply AI-assisted programming to implement and optimize sorting and searching algorithms. * Compare different algorithms in terms of efficiency and use cases. * Understand how AI tools can suggest optimized code and complexity improvements.   **Task Description #1 (Sorting – Merge Sort Implementation)**   * Task: Use AI to generate a Python program that implements the Merge Sort algorithm. * Instructions:   + Prompt AI to create a function merge\_sort(arr) that sorts a list in ascending order.   + Ask AI to include time complexity and space complexity in the function docstring.   + Verify the generated code with test cases. * Expected Output:   + A functional Python script implementing Merge Sort with proper documentation.   **Prompt#1:**   * Write a Python program that implements merge sort with a merge\_sort(arr) function. Include time and space complexity in the function docstring, and add test cases with dynamic user input.   **Code#1:**      **Observation#1:**   * This Python code implements the merge sort algorithm, a comparison-based sorting algorithm known for its time complexity of O(n log n) in all cases. The code includes a merge\_sort function for recursive splitting and sorting, and a merge function to combine the sorted sub-arrays. It also features a run\_merge\_sort\_tests function that allows for dynamic input of either purely numeric or purely string data.   **Task Description #2 (Searching – Binary Search with AI Optimization)**   * Task: Use AI to create a binary search function that finds a target element in a sorted list. * Instructions:   + Prompt AI to create a function binary\_search(arr, target) returning the index of the target or -1 if not found.   + Include docstrings explaining best, average, and worst-case complexities.   + Test with various inputs. * Expected Output:   + Python code implementing binary search with AI-generated comments and docstrings.     **Prompt#2:**   * Write a Python program with a binary\_search(arr, target) function that returns the index of a target in a sorted list or -1 if not found, including time complexity in the docstring and dynamic user input.   **Code#2:**      **Observation#2:**   * This Python code implements the binary search algorithm, an efficient search algorithm for sorted lists with a time complexity of O(log n) in average and worst-case scenarios, and O(1) in the best case. The code defines a binary\_search function that takes a sorted list and a target element as input, returning the index of the target if found or -1 otherwise. It includes a run\_binary\_search\_tests function to allow users to input a sorted list and a target element dynamically, displaying the search result and the time taken.   **Task Description #3 (Real-Time Application – Inventory Management System)**   * Scenario: A retail store’s inventory system contains thousands of products, each with attributes like product ID, name, price, and stock quantity. Store staff need to:   1. Quickly search for a product by ID or name.   2. Sort products by price or quantity for stock analysis. * Task:   1. Use AI to suggest the most efficient search and sort algorithms for this use case.   2. Implement the recommended algorithms in Python.   3. Justify the choice based on dataset size, update frequency, and performance requirements. * Expected Output:   1. A table mapping operation → recommended algorithm → justification.   2. Working Python functions for searching and sorting the inventory.   **Prompt#3:**   * Write a Python code to manage a retail store inventory that allows dynamic searching by ID or name and sorting by price or quantity. Use efficient algorithms like hash tables for ID lookups and binary search for name searches, displaying results with time and space complexity explanations.   **Code#3:**        **Observation#3:**   * The program efficiently manages a retail store inventory, allowing dynamic searches by ID or name and sorting by price or quantity. It uses hash tables for constant-time ID lookups and binary search for name searches, providing fast, reliable results with clear time and space complexity.   ✅ Deliverables (For All Tasks)   1. AI-generated prompts for code and test case generation. 2. At least 3 assert test cases for each task. 3. AI-generated initial code and execution screenshots. 4. Analysis of whether code passes all tests. 5. Improved final version with inline comments and explanation. 6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.   Top of Form | | | | | | Week6 - Monday |  |