|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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) | | | | | | | | | |
|  | | | | | | | | | |
|  | **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:**  Generate a Python program with a function merge\_sort(arr) that sorts a list using Merge Sort. Include time and space complexity in the docstring.  **Code:**      **Code Explanation:**  1.merge\_sort function splits the list into halves until one element remains.  **2.Docstring**:Explains the purpose of Merge Sort..Lists **time complexity** Lists **space complexity.**  **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:**  Generate a Python program with a function binary\_search(arr, target) that searches for a target element in a sorted list and returns its index, or -1 if not found. Include a docstring explaining best, average, and worst-case time complexities.  **Code:**      **Code explanation:**  1.binary\_search takes a sorted list and a target value.  2.Uses left and right pointers to search within the list.  3.Finds the middle index and compares it with the target.  4.If target is smaller → search left half.  5.If target is larger → search right half.  **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:**  Generate a Python functions to search (by ID or name) and sort (by price or quantity) a store inventory.  **Code:**          **Code Explanation:**   1. Each product is a dictionary with id, name, price, and quantity. 2. search\_by\_id uses a dictionary for fast O(1) lookup by product ID. 3. search\_by\_name uses linear search to find a product by name 4. sort\_by\_price sorts products by price using Python’s built-in sorted(). 5. sort\_by\_quantity sorts products by quantity using sorted(). 6. Prints results of search and sorted lists to verify correctness.   ✅ 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 |  |