|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 - Thursday | **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 1: Implementing Bubble Sort with AI Comments**   * **Task**: Write a Python implementation of **Bubble Sort**. * **Instructions**:   + Students implement Bubble Sort normally.   + Ask AI to generate **inline comments explaining key logic** (like swapping, passes, and termination).   + Request AI to provide **time complexity analysis**. * **Expected Output**:   + A Bubble Sort implementation with AI-generated explanatory comments and complexity analysis.     **Task 2: Optimizing Bubble Sort → Insertion Sort**   * **Task**: Provide Bubble Sort code to AI and ask it to suggest a **more efficient algorithm** for partially sorted arrays. * **Instructions**:   + Students implement Bubble Sort first.   + Ask AI to suggest an alternative (Insertion Sort).   + Compare performance on nearly sorted input. * **Expected Output**:   + Two codes (Bubble Sort + Insertion Sort).   + AI explanation of why Insertion Sort is more efficient for partially sorted data.        * **TTask 3: Binary Search vs Linear Search** * **Task**: Implement both **Linear Search** and **Binary Search**. * **Instructions**:   + Use AI to generate docstrings and performance notes.   + Test both algorithms on sorted and unsorted data.   + Ask AI to explain when Binary Search is preferable. * **Expected Output**:   + Two implementations with docstrings.   + A student observation table comparing performance (Linear vs Binary Search).     Top of Form | | | | | | Week6 - Thursday |  |
|  |  | **Task 4:** Quick Sort and Merge Sort Comparison   * **Task:** Implement Quick Sort and Merge Sort using recursion**.** * **Instructions:**   + Provide AI with partially completed functions for recursion.   + Ask AI to complete the missing logic and add docstrings.   + Compare both algorithms on random, sorted, and reverse-sorted lists. * **Expected Output:**   + Working Quick Sort and Merge Sort implementations.   + AI-generated explanation of average, best, and worst-case complexities. | | | | | |  |  |
|  |  | **Task 5:** AI-Suggested Algorithm Optimization   * **Task:** Give AI a naive algorithm (e.g., O(n²) duplicate search). * **Instructions:**   + Students write a brute force duplicate-finder.   + Ask AI to optimize it (e.g., by using sets/dictionaries with O(n) time).   + Compare execution times with large input sizes. * **Expected Output:**   + Two versions of the same algorithm (brute force + optimized).   + AI explanation of how complexity was improved. | | | | | |  |  |