|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | **Week1 - Monday** | **Time(s)** | | 24CSBTB01 To 24CSBTB39 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | | |
| **Assignment Number:1.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | Lab 1: Environment Setup – GitHub Copilot and VS Code Integration  **Lab Objectives:**   * To install and configure GitHub Copilot in Visual Studio Code. * To explore AI-assisted code generation using GitHub Copilot. * To analyze the accuracy and effectiveness of Copilot's code suggestions. * To understand prompt-based programming using comments and code context   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Set up GitHub Copilot in VS Code successfully. * Use inline comments and context to generate code with Copilot. * Evaluate AI-generated code for correctness and readability. * Compare code suggestions based on different prompts and programming styles.   Task 0   * Install and configure GitHub Copilot in VS Code. Take screenshots of each step.   Expected Output   * Install and configure GitHub Copilot in VS Code. Take screenshots of each step.     Task 1: Factorial without Functions   * Description: Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions (using loops directly in the main code). * Expected Output:   + A working program that correctly calculates the factorial for user-provided input.   + Screenshots of the code generation process.     Task 2: Improving Efficiency   * Description: Examine the Copilot-generated code from Task 1 and demonstrate how its efficiency can be improved (e.g., removing unnecessary variables, optimizing loops). * Expected Output:   + Original and improved versions of the code.   + Explanation of how the improvements enhance performance.      * Loop runs efficiently from 1 to n, multiplying as it goes. * Input handling is simple and direct * No unnecessary variables—just n, factorial, and i.   Task 3: Factorial with Functions   * Description: Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function. * Expected Output:   + Correctly working factorial function with sample outputs.   + Documentation of the steps Copilot followed to generate the function.     Task 4: Comparative Analysis – With vs Without Functions   * Description: Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic, reusability, and execution. * Expected Output:   + A comparison table or short report explaining the differences.     The **function-based implementation** offers superior structure, allowing the factorial logic to be abstracted and reused in other contexts. This makes it suitable for more complex programs, encourages cleaner code practices, and simplifies testing. On the other hand, the **loop-only version** is straightforward and perfectly fine for quick, one-off calculations—but it sacrifices modularity and flexibility.  💡 Want to see how recursion or memoization changes this game entirely? We can dive into that next.  Task 5: Iterative vs Recursive Factorial   * Description: Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function. * Expected Output:   + Two correct implementations.   + A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.     **Submission Requirements**   1. Generate code for each task with comments. 2. Screenshots of Copilot suggestions. 3. Comparative analysis reports (Task 4 and Task 5). 4. Sample inputs/outputs demonstrating correctness.   **Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Successful Setup of Copilot | 0.5 | | Comparative Analysis – With vs Without Functions | 1 | | Iterative vs Recursive Factorial | 1 | | **Total** | **2.5 Marks** | | | | | | | Week1 - Monday |  |