|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | Dr. Rishabh Mittal | | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | Mr. S Naresh Kumar | | Ms. B. Swathi | | Dr. Sasanko Shekhar Gantayat | | Mr. Md Sallauddin | | Dr. Mathivanan | | Mr. Y Srikanth | | Ms. N Shilpa | | Dr. Rishabh Mittal (Coordinator) | | Dr. R. Prashant Kumar | | Mr. Ankushavali MD | | Mr. B Viswanath | | Ms. Rapelly Nandini | | Ms. A. Anitha | | Ms. M.Madhuri | | Ms. Katherashala Swetha | | Ms. Velpula sumalatha | | Mr. Bingi Raju | | | | | | |
| **CourseCode** | | | 23CS002PC304 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | III/II | **Regulation** | | R23 | | | |
| **Date and Day**  **of Assignment** | | | **Week1 - Wednesday** | **Time(s)** | | 23CSBTB01 To 23CSBTB52 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | | |
| **Assignment Number:1.3**(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 + Understanding AI-assisted Coding Workflow*  **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: AI-Generated Logic Without Modularization (Fibonacci Sequence Without Functions)   * **Scenario**   You are asked to write a quick numerical sequence generator for a learning platform prototype.   * **Task Description**   Use GitHub Copilot to generate a Python program that:   * + Prints the Fibonacci sequence up to *n* terms   + Accepts user input for *n*   + Implements the logic directly in the main code   + Does not use any user-defined functions * **Expected Output**   + Correct Fibonacci sequence for given *n*   + Screenshot(s) showing Copilot-generated suggestions   + Sample inputs and outputs   + PROMT: Write python code to display the Fibonacci sequence upto n terms   + CODE AND OUTPUT:     EXPLANATION:  The program takes an input nterms to decide how many Fibonacci numbers to display. Each new Fibonacci number is calculated by adding the previous two numbers and printed until the count reaches nterms.  Task 2: AI Code Optimization & Cleanup (Improving Efficiency)   * **Scenario**   The prototype will be shared with other developers and needs optimization.   * **Task Description**   + Examine the Copilot-generated code from Task 1 and improve it by:   + Removing redundant variables   + Simplifying loop logic   + Avoiding unnecessary computations   + Use Copilot prompts such as:     - *“Optimize this Fibonacci code”*     - *“Simplify variable usage”*   Hint: Prompt Copilot with phrases like *“optimize this code”*, *“simplify logic”*, or *“make it more readable”*   * **Expected Output**   + Original vs improved code   + Written explanation of:     - What was inefficient     - How the optimized version improves performance and readability * SOLUTION: * PROMPT:   Write python code to display the Fibonacci sequence using simplify logic.  🡪CODE AND OUTPUT:    **EXPLANATION:**  **The earlier code generates and prints Fibonacci numbers step by step using a while loop and conditions, while the present code uses a function with a for loop to store the sequence in a list and print it all at once.**  Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions)   * **Scenario**   The Fibonacci logic is now required in multiple modules of an application.   * **Task Description**   Use GitHub Copilot to generate a function-based Python program that:   * + Uses a user-defined function to generate Fibonacci numbers   + Returns or prints the sequence up to *n*   + Includes meaningful comments (AI-assisted) * **Expected Output**   + Correct function-based Fibonacci implementation   + Screenshots documenting Copilot’s function generation   + Sample test cases with output * SOLUTION: * PROMPT:   Write a python code to display fiboncci using functions.  🡪CODE AND OUTPUT:    🡪EXPLANATION:  This code uses a function to generate the Fibonacci sequence up to a given number of terms.It checks for invalid or small inputs and then calculates each term using a for loop. The generated Fibonacci numbers are stored in a list and printed as output.  Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code   * **Scenario**   You are participating in a code review session.   * **Task Description**   Compare the Copilot-generated Fibonacci programs:   * + Without functions (Task 1)   + With functions (Task 3)   + Analyze them in terms of:     - Code clarity     - Reusability     - Debugging ease     - Suitability for larger systems * **Expected Output**   Comparison table or short analytical report  🡪PROMPT:  Write python code to display Fibonacci series using functions and also by not using functions.  🡪CODE AND OUTPUT:      🡪EXPLANATION:  Using functions organizes the Fibonacci logic into a reusable and clean block of code, making it easier to understand and maintain, while not using functions places all logic in the main program, which is simpler for beginners but harder to reuse and manage in larger programs.  Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for Fibonacci Series)   * **Scenario**   Your mentor wants to assess AI’s understanding of different algorithmic paradigms.   * **Task Description**   Prompt GitHub Copilot to generate:  An iterative Fibonacci implementation  A recursive Fibonacci implementation   * **Expected Output**   + Two correct implementations   + Explanation of execution flow for both   + Comparison covering:     - Time and space complexity     - Performance for large *n*     - When recursion should be avoided   **🡪SOLUTION:**  **🡪PROMPT:**  **Write a python code to display an iterative fibonacci implementation.**  **Write a python code to display a recursive fibonacci implementation,**  **🡪CODE AND OUTPUT:**      **🡪EXPLANATION:**  **Iterative Fibonacci uses a loop and computes values efficiently with O(n) time and O(1) space.Recursive Fibonacci uses repeated function calls, which is simple but has O(2ⁿ) time and O(n) space.Iterative execution flows in a single forward loop, while recursive execution breaks into smaller calls.For large n, iterative performs much faster and avoids stack overflow.Recursion should be avoided when performance, memory usage, or large input sizes matter.**  **Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.** | | | | | | Week1 - Monday |  |