|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 - Tuesday** | **Time(s)** | | 23CSBTB01 To 23CSBTB52 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | | |
| **Assignment Number:1.2**(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.     Task 1: AI-Generated Logic Without Modularization (Factorial without Functions)   * **Scenario**   You are building a **small command-line utility** for a startup intern onboarding task. The program is simple and must be written quickly without modular design.  **Prompt:** Write a Python program to calculate factorial without using functions.    **Description:** This task generates a factorial program without using any functions.The logic is implemented directly using loops and variables.  **Task 2: AI Code Optimization & Cleanup (Improving Efficiency)**   * **Scenario**   Your team lead asks you to **review AI-generated code** before committing it to a shared repository.   * **Prompt:** Optimize this factorial code and make it more readable.     **Description:** This task improves the AI-generated factorial code by optimizing the logic.Readability and efficiency of the program are enhanced.  **Task 3: Modular Design Using AI Assistance (Factorial with Functions)**   * **Scenario**   The same logic now needs to be reused in **multiple scripts**.   * **Prompt**: Write a Python factorial program using a function.     **Description:** This task converts the factorial logic into a modular program using a function.The same logic can be reused in multiple programs**.**  Task 4: Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions)   | **Criteria** | **Procedural Code (Without Functions)** | **Modular Code (With Functions)** | | --- | --- | --- | | **Logic Clarity** | Logic is written in one block, which is easy for small programs but becomes confusing as code grows | Logic is separated into functions, making it cleaner and easier to understand | | **Reusability** | Code cannot be reused easily; logic must be rewritten in every program | Functions can be reused in multiple programs without rewriting | | **Debugging Ease** | Harder to debug because all logic is in one place | Easier to debug since errors can be traced to a specific function | | **Suitability for Large Projects** | Not suitable for large projects as code becomes lengthy and messy | Very suitable for large projects due to structured and organized code | | **AI Dependency Risk** | Higher risk, as developers may copy full code without understanding | Lower risk, as modular design encourages understanding each function |   **Description:** This task compares factorial programs written with and without functions.The comparison focuses on clarity, reusability, and scalability.  **Task 5: AI-Generated Iterative vs Recursive Thinking**   * **Scenario**   Your mentor wants to test how well AI understands different computational paradigms.   * **Prompt:** Write iterative and recursive factorial programs in Python.     **Description:** This task generates iterative and recursive factorial programs. Both approaches are compared to understand their behavior and performance. | | | | | | Week1 - –onday |  |