|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | 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 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day of Assignment** | | | Week4 - Thursday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto Batches** | |  | | | |
| **AssignmentNumber:7.4**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
| CH.Divya | | | | | | | | | |
| 2403a52409 | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | | ***ExpectedTi me***  ***to***  ***complete*** |
|  | 1 | Lab 7: Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs  Lab Objectives:   * To identify and correct syntax, logic, and runtime errors in Python programs using AI tools. | | | | | | | Week4 - Thursday |

|  |  |  |
| --- | --- | --- |
|  | * To understand common programming bugs and AI-assisted debugging suggestions. * To evaluate how AI explains, detects, and fixes different types of coding errors. * To build confidence in using AI to perform structured debugging practices. Lab Outcomes (LOs):   After completing this lab, students will be able to:   * Use AI tools to detect and correct syntax, logic, and runtime errors. * Interpret AI-suggested bug fixes and explanations. * Apply systematic debugging strategies supported by AI-generated insights. * Refactor buggy code using responsible and reliable programming patterns.   **Task Description #1:**   * Introduce a buggy Python function that calculates the factorial of a number using recursion. Use Copilot or Cursor AI to detect and fix the logical or syntax errors.   **Expected Outcome #1:**   * Copilot or Cursor AI correctly identifies missing base condition or incorrect recursive call and suggests a functional factorial implementation.     **Task Description #2:**   * Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detect the issue and fix the code for consistent sorting**. Expected Outcome #2:** * AI detects the type inconsistency and either filters or converts list elements, ensuring   successful sorting without a crash. |  |

|  |  |  |
| --- | --- | --- |
|  | **Task Description #3:**   * Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).   **Expected Outcome #3:**   * AI refactors the code to use a context manager, preventing resource leakage and runtime warnings.   **Task Description #4:**   * Provide a piece of code with a ZeroDivisionError inside a loop. Ask AI to add error handling using try-except and continue execution safely.   **Expected Outcome #4:**   * Copilot adds a try-except block around the risky operation, preventing crashes and printing a meaningful error message. |  |

|  |  |  |
| --- | --- | --- |
|  | OBSERVATION:   * The original code attempted to divide 100 by each number in the list, which caused a ZeroDivisionError when it encountered 0. * The revised version uses a try-except block to catch this specific error, allowing the program to continue executing without interruption. * Instead of crashing, the program now prints a clear message: "Cannot divide by zero. Skipping value: 0", which improves user experience and debugging. * The loop continues smoothly after handling the error, demonstrating   **robustness** and **fault tolerance** in the code design. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Task Description #5:**   * Include a buggy class definition with incorrect init parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.   **Expected Outcome #5:**   * Copilot identifies mismatched parameters or missing self references and rewrites the class with accurate initialization and usage.   **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** |  |
| Logic | 0.5 |
| Type mismatch in list elements during sorting | 0.5 |
| Resource | 0.5 |
| Runtime | 0.5 |
| Syntax | 0.5 |
| **Total** | **2.5 Marks** |