|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 - Wednesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:7.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 7: AI-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. * 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**   * Paste a function with a missing colon (add(a, b)), and let AI fix the syntax error.     **Expected Output#1**   * Corrected function with syntax fix   **Prompt:**   * Fix the syntax error in this Python code:   **Generated Code:**    **Explanation:**   * AI detected that the function header was missing a colon (:) after the parameter list. * It corrected the syntax and provided a proper structure. * The corrected function now executes successfully and returns the sum.   **Example Output:**    **Task Description#2 (Loops)**   * Identify and fix a logic error in a loop that causes infinite iteration.     **Expected Output#2**   * AI fixes increment/decrement error   **Prompt:**   * The following code runs infinitely. Identify and fix the logic error.   **Generated Code:**    **Explanation:**   * The original loop had no increment statement, leading to an infinite loop. * AI added i += 1 to update the loop variable. * Now the program executes five times and terminates correctly.   **Example Output:**    **Task Description#3**   * Debug a runtime error caused by division by zero. Let AI insert try-except.     **Expected Output#3**   * Corrected function with safe error handling   **Prompt:**   * Fix the runtime error in the following code that crashes due to division by zero.   **Generated Code:**    **Explanation:**   * AI recognized that dividing by zero causes a runtime exception. * It added a try-except block to handle ZeroDivisionError, preventing the program from * crashing and providing a meaningful message.   **Example Output:**    **Task Description#4**   * Provide a faulty class definition (missing self in parameters). Let AI fix it     **Expected Output#4**   * Correct \_\_init\_\_() method and explanation   **Prompt:**   * Fix the missing self parameter in the following Python class:   **Generated Code:**    **Explanation:**   * The original code missed the self parameter in \_\_init\_\_(). * AI added self as the first argument in the constructor and correctly assigned instance variables. * Now the class creates objects successfully**.**   **Example Output:**    **Task Description#5**   * Access an invalid list index and use AI to resolve the Index Error.     **Expected Output#5**   * AI suggests checking length or using safe access logic   **Prompt:**   * Fix the following Python code that raises an IndexError.   **Generated Code:**    **Explanation:**   * AI added a length check before accessing the list element. * This prevents IndexError and provides a clear error message if the index is invalid.   **Example Output:**    **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** | | --- | --- | | Identification of bugs | 0.5 | | Application of AI-suggested fixes | 0.5 | | Explanation and understanding of errors | 0.5 | | Corrected code functionality | 0.5 | | Report structure and reflection | 0.5 | | **Total** | **2.5 Marks** | | | | | | | Week4 - Wednesday |  |