|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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. Sujitha Reddy | | 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** | | | **Week4 – Monday** | **Time(s)** | | 23CSBTB01 To 23CSBTB52 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | | |
| **Assignment Number: 7.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***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. * 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** (Syntax Errors – Missing Parentheses in Print Statement)  Task: Provide a Python snippet with a missing parenthesis in a print statement (e.g., print "Hello"). Use AI to detect and fix the syntax error.  # Bug: Missing parentheses in print statement  def greet():  print "Hello, AI Debugging Lab!"  greet()  Requirements:   * Run the given code to observe the error. * Apply AI suggestions to correct the syntax. * Use at least 3 assert test cases to confirm the corrected code works.   Expected Output #1:   * Corrected code with proper syntax and AI explanation.   **Task Description #2** (Incorrect condition in an If Statement)  Task: Supply a function where an if-condition mistakenly uses = instead of ==. Let AI identify and fix the issue.  # Bug: Using assignment (=) instead of comparison (==)  def check\_number(n):  if n = 10:  return "Ten"  else:  return "Not Ten"  Requirements:   * Ask AI to explain why this causes a bug. * Correct the code and verify with 3 assert test cases.   Expected Output #2:   * Corrected code using == with explanation and successful test execution.   **Task Description #3** (Runtime Error – File Not Found)  Task: Provide code that attempts to open a non-existent file and crashes. Use AI to apply safe error handling.  # Bug: Program crashes if file is missing  def read\_file(filename):  with open(filename, 'r') as f:  return f.read()  print(read\_file("nonexistent.txt"))  Requirements:   * Implement a try-except block suggested by AI. * Add a user-friendly error message. * Test with at least 3 scenarios: file exists, file missing, invalid path.   Expected Output #3:   * Safe file handling with exception management.   **Task Description #4** (Calling a Non-Existent Method)  Task: Give a class where a non-existent method is called (e.g., obj.undefined\_method()). Use AI to debug and fix.  # Bug: Calling an undefined method  class Car:  def start(self):  return "Car started"  my\_car = Car()  print(my\_car.drive()) # drive() is not defined  Requirements:   * Students must analyze whether to define the missing method or correct the method call. * Use 3 assert tests to confirm the corrected class works.   Expected Output #4:   * Corrected class with clear AI explanation.   **Task Description #5** (TypeError – Mixing Strings and Integers in Addition)  Task: Provide code that adds an integer and string ("5" + 2) causing a TypeError. Use AI to resolve the bug.  # Bug: TypeError due to mixing string and integer  def add\_five(value):  return value + 5  print(add\_five("10"))  Requirements:   * Ask AI for two solutions: type casting and string concatenation. * Validate with 3 assert test cases.   Expected Output #5:   * Corrected code that runs successfully for multiple inputs.   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 | | | | | | Week4 - Monday |  |