|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | 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) | | | | | | |
| **Course Code** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week5 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **AssignmentNumber: 9.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 9 – Code Review and Quality: Using AI to improve code quality and readability**  **Lab Objectives**   * Inline comments * Docstrings * Auto-documentation tools * AI-assisted summarization   **Task Description #1** (AI-Assisted Bug Detection)  **Scenario:** A junior developer wrote the following Python function to calculate factorials:  def factorial(n):  result = 1  for i in range(1, n):  result = result \* i  return result   * Run the code and test it with factorial(5)   (expected output = 120).   * Use AI (prompting) to review this code and identify the bug. * Ask AI to suggest corrections and rewrite the code. * Compare AI’s corrected code with your own fix.   #Prompt : correct the given code and give me output  #Correction of code using AI:    #Comparasion :  That's great that you corrected the indentation in the for loop! That was definitely one of the issues in the original code.  As we discussed, the other crucial correction for calculating the factorial correctly was changing range(1, n) to range(1, n + 1) so that the loop includes the number n itself in the multiplication.  **Task Description #2** (Improving Readability & Documentation)  **Scenario:** The following code works but is poorly written:  def calc(a,b,c):  if c=="add":  return a+b  elif c=="sub":  return a-b  elif c=="mul":  return a\*b  elif c=="div":  return a/b   * Use AI to review this code for readability, naming, and documentation issues. * Prompt AI to rewrite the code with: * Clear function & variable names. * Proper docstrings. * Exception handling for division by zero.   Compare the before-and-after versions to evaluate AI’s contribution.  #prompt : rewrite the code by including clear function and variable names , proper docstrings, exception handlings for division by zero.also make itshort #Correction of code :    #Comparison :   1. **Clearer Names:** The function name is now perform\_calculation instead of calc, and the parameters are named num1, num2, and operation which are more descriptive than a, b, and c. 2. **Docstrings:** The refactored code includes a comprehensive docstring that explains what the function does, its arguments, what it returns, and what exceptions it might raise. This makes the code much easier to understand and use. 3. **Exception Handling for Division by Zero:** The refactored code specifically includes a try...except ZeroDivisionError block within the division case. This prevents the program from crashing when attempting to divide by zero and provides a user-friendly error message. 4. **Handling Invalid Operations:** The refactored code includes an else block that raises a ValueError for any operation that is not 'add', 'sub', 'mul', or 'div'. This makes the function more robust to invalid input. 5. **Example Usage:** We added example usage to demonstrate how to call the function and how the exception handling works.   In summary, the refactored code is more readable, maintainable, and robust due to the inclusion of clear names, docstrings, and comprehensive exception handling.  **Task Description #3** (Enforcing Coding Standards)  **Scenario:** A team project requires following PEP8 style guide. One developer submits:  def Checkprime(n):  for i in range(2,n):  if n%i==0:  return False  return True+   * Run this code and verify correctness. * Use AI to perform a code quality review for PEP8 compliance. * Prompt AI to return a refactored version with proper indentation, spacing, and naming conventions. * Discuss how automated AI review can save time in large-scale projects.   Prompt : rewrite the code such that it returns the refactored version with proper indentation, spacing, and naming conventions.(take input from user)  #Ai corrected one :    #comparison:   1. **Function Name:** The original function name Checkprime has been changed to is\_prime. This is a more descriptive and standard naming convention in Python for functions that return a boolean value (True or False). 2. **Variable Names:** The variable names inside the function have been improved. num is changed to number, which is more explicit. The loop variable x is changed to i, which is a common and understandable convention for loop indices. 3. **Docstring:** The refactored version includes a docstring. This is a crucial addition as it explains what the function does, its arguments, and what it returns. This makes the code much easier to understand and use. 4. **Indentation and Spacing:** While the original code you provided already had decent indentation, the refactored version ensures consistent spacing around operators and in the function definition, contributing to better readability.   In summary, the refactored code is more readable, maintainable, and follows better Python coding practices through improved naming, the addition of a docstring, and consistent formatting.  **Task Description #4** (AI as a Code Reviewer in Real Projects)  **Scenario:** You are part of a GitHub project. A teammate submits this pull request:  def processData(d):  return [x\*2 for x in d if x%2==0]   * Review this function manually for readability, reusability, and edge cases. * Use AI to generate a code review comment, focusing on: * Naming conventions. * Input validation (e.g., what if d is not a list?). * Adding type hints. * Modify the function based on AI’s suggestions. * Write a short reflection: *Would you trust AI as a standalone reviewer, or only as a support tool? Why?*   *#prompt : Rewrite the code by naming the conventions and also giving input validation ,and adding type hints in it . give along with output*  *#corrected code:*    #Reflection :  I would consider AI just as an supporting tool , since it can help us generate the code but it cant generate it effectively unless the user specifices the condition requirements of what code he needs and the format of output they generally need . Even after specifiying all these needs, the ai sometimes cannot generate code according to user requirements due to lack of specifications . | | | | | | Week5 - Monday |  |