|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.       Data Analysis Key Findings   * The initial factorial function correctly calculated the factorial of 5 as 120. * No bug was identified in the provided factorial function for the input 5, as the actual result matched the expected result. * The revised code, incorporating an iterative approach and handling negative inputs, also correctly calculated the factorial of 5 as 120, confirming the fix.   **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 the code with: * Clear function & variable names. * Proper docstrings. * Exception handling for division by zero. * Compare the before-and-after versions rewrite to evaluate AI’s contribution.       The AI made the following improvements:   1. **Clearer Function and Variable Names:** The function name was changed from calc to perform\_calculation, and the parameters from a, b, c to num1, num2, and operation. This makes the code's purpose and inputs much more understandable. 2. **Proper Docstrings:** A comprehensive docstring was added to explain what the function does, its arguments, what it returns, and what exceptions it might raise. This is crucial for code documentation and maintainability. 3. **Exception Handling for Division by Zero:** The refactored code explicitly checks if num2 is zero before performing division and raises a ValueError if it is. This prevents a ZeroDivisionError and makes the function more robust.   **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.     Data Analysis Key Findings   * The original Checkprime function correctly identified prime and non-prime numbers in the provided examples. * The original Checkprime function violated PEP8 guidelines by using CamelCase for the function name (Checkprime) and lacking proper spacing around operators (\*\*, +) and after the comma in the range() function call. * The refactored is\_prime function adheres to PEP8 standards with a snake\_case name, a docstring, and appropriate spacing. * Automated AI code review tools save time by automating routine checks and improve code quality by enforcing standards and identifying potential issues.   **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?*     Data Analysis Key Findings   * The initial manual code review could not be completed because the processData function definition was missing. * AI was successfully used to generate instructions for a code review focusing on naming conventions, input validation, and type hints. * A refactored process\_data function was created based on the hypothetical AI suggestions, incorporating snake\_case naming, type hints (list and list[int]), input validation for lists using isinstance, and a docstring. * The refactored function correctly handles valid list inputs and raises a TypeError for invalid non-list inputs. * A reflection was generated concluding that AI is a valuable supporting tool in code review for handling routine checks and enforcing standards, but should not be a standalone reviewer due to its lack of understanding of complex logic, business requirements, and architectural nuances. | | | | | | Week5 - Monday |  |