|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:10.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 10 – Code Review and Quality: Using AI to Improve Code Quality and Readability**  **Lab Objectives**   * Use AI for automated code review and quality enhancement. * Identify and fix syntax, logical, performance, and security issues in Python code. * Improve readability and maintainability through structured refactoring and comments. * Apply prompt engineering for targeted improvements. * Evaluate AI-generated suggestions against PEP 8 standards and software engineering best practices   **Task Description #1 – Syntax and Logic Errors**  Task: Use AI to identify and fix syntax and logic errors in a faulty Python script.  Sample Input Code:  # Calculate average score of a student  def calc\_average(marks):  total = 0  for m in marks:  total += m  average = total / len(marks)  return avrage # Typo here  marks = [85, 90, 78, 92]  print("Average Score is ", calc\_average(marks)  Expected Output:   * Corrected and runnable Python code with explanations of the fixes.   **PROMPT:**  Identify and fix syntax and logic errors in the below faulty Python script.    # Calculate average score of a student  def calc\_average(marks):  total = 0  for m in marks:  total += m  average = total / len(marks)  return avrage # Typo here  marks = [85, 90, 78, 92]  print("Average Score is ", calc\_average(marks)  **CODE:**      **OUTPUT:**      **EXPLANATION:**   * **Function Name Fix:** Changed calc\_averge to calc\_average for correct spelling and readability. * **Variable Name Fix:** Changed avrage to average to match the calculated variable. * **Logic Check:** The average is computed correctly using total / len(marks).   **Task Description #2 – PEP 8 Compliance**  Task: Use AI to refactor Python code to follow PEP 8 style guidelines.  Sample Input Code:  def area\_of\_rect(L,B):return L\*B  print(area\_of\_rect(10,20))  Expected Output:   * Well-formatted PEP 8-compliant Python code.   **PROMPT:**  Refactor the below Python code to follow PEP 8 style guidelines.  def area\_of\_rect(L,B):return L\*B  print(area\_of\_rect(10,20))  **CODE:**    **OUTPUT:**    **EXPLANATION:**   * **Function Naming:** Changed area\_of\_rect to area\_of\_rectangle — descriptive and PEP 8-compliant. * **Variable Naming:** Changed L to length and B to breadth — lowercase and meaningful. * **Spacing:** Added spaces after commas. * **Formatting:** Placed the return statement on its own line to improve readability and conform to PEP 8 style.   **Task Description #3 – Readability Enhancement**  Task: Use AI to make code more readable without changing its logic**.**  Sample Input Code:  def c(x,y):  return x\*y/100  a=200  b=15  print(c(a,b))  Expected Output:   * Python code with descriptive variable names, inline comments, and clear formatting.   **PROMPT:**  Make the below code more readable without changing its logic**.**  def c(x,y):  return x\*y/100  a=200  b=15  print(c(a,b))  **CODE:**    **OUTPUT:**    **EXPLANATION:**  **Improvements:**   1. **Function name** → c → changed to calculate\_percentage (descriptive). 2. **Parameter names** → x, y → changed to value, percentage. 3. **Variables** → a, b → changed to amount, rate. 4. **Docstring** → added explanation (professional & PEP 257). 5. **Comments** → explain what each step does. 6. **Formatting** → consistent spacing and readability.   **Task Description #4 – Refactoring for Maintainability**  Task: Use AI to break repetitive or long code into reusable functions.  Sample Input Code:  students = ["Alice", "Bob", "Charlie"]  print("Welcome", students[0])  print("Welcome", students[1])  print("Welcome", students[2])  Expected Output:   * Modular code with reusable functions.   **PROMPT:**  Break the below repetitive or long code into reusable functions.  students = ["Alice", "Bob", "Charlie"]  print("Welcome", students[0])  print("Welcome", students[1])  print("Welcome", students[2])  **CODE:**    **OUTPUT:**    **EXPLANATION:**   * **welcome\_student(name)** → prints welcome message for one student. * **welcome\_all\_students(student\_list)** → loops through list and calls welcome\_student for each. * students = ["Alice", "Bob", "Charlie"] → list of names. * **welcome\_all\_students(students)** → prints welcome messages for all.   **Task Description #5 – Performance Optimization**  Task: Use AI to make the code run faster.  Sample Input Code:  # Find squares of numbers  nums = [i for i in range(1,1000000)]  squares = []  for n in nums:  squares.append(n\*\*2)  print(len(squares))  Expected Output:   * Optimized code using list comprehensions or vectorized operations.   **PROMPT:**  Make the below code run faster.  # Find squares of numbers  nums = [i for i in range(1,1000000)]  squares = []  for n in nums:  squares.append(n\*\*2)  print(len(squares))  **CODE:**      **OUTPUT:**    **EXPLANATION:**  Creates a **list in memory** holding all 1,000,000 squared numbers.  **Pros:**  Very fast (optimized in C).  Easy to access elements later (squares[5], len(squares), slicing, etc.).  **Cons:**  Uses **a lot of memory** (all 1M numbers stored at once).  **Task Description #6 – Complexity Reduction**  Task: Use AI to simplify overly complex logic.  Sample Input Code:  def grade(score):  if score >= 90:  return "A"  else:  if score >= 80:  return "B"  else:  if score >= 70:  return "C"  else:  if score >= 60:  return "D"  else:  return "F"  Expected Output:   * Cleaner logic using elif or dictionary mapping.   **PROMPT:**  Simplify overly complex logic.  def grade(score):  if score >= 90:  return "A"  else:  if score >= 80:  return "B"  else:  if score >= 70:  return "C"  else:  if score >= 60:  return "D"  else:  return "F"  **CODE:**    **OUTPUT:**    **EXPLANATION:**   * map score thresholds → grades in a dictionary. * Loop through the mapping and return the first grade that fits. * Cleaner, easier to extend (just add {50: "E"} if needed). | | | | | | Week5 - Monday |  |