|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 - Thursday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **AssignmentNumber:10.4**(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 1: Syntax and Error Detection**  **Task:** Identify and fix syntax, indentation, and variable errors in the given script.  # buggy\_code\_task1.py  def add\_numbers(a, b)  result = a + b  return reslt  print(add\_numbers(10 20))  **Expected Output**:   * Corrected code with proper syntax (: after function, fixed variable name, corrected function call). * AI should explain what was fixed.   **Prompt:**    **Code:**      **Output:**    **Task 2: Logical and Performance Issue Review**  **Task**: Optimize inefficient logic while keeping the result correct.  # buggy\_code\_task2.py  def find\_duplicates(nums):  duplicates = []  for i in range(len(nums)):  for j in range(len(nums)):  if i != j and nums[i] == nums[j] and nums[i] not in duplicates:  duplicates.append(nums[i])  return duplicates  numbers = [1,2,3,2,4,5,1,6,1,2]  print(find\_duplicates(numbers))  **Expected Output**:   * More efficient duplicate detection (e.g., using sets). * AI should explain the optimization.   **Prompt:**    **Code:**    **Output:**    **Explanation:**  Efficiency Comparison:  - Original `find\_duplicates` function had a time complexity of approximately O(n^3) due to nested loops and list lookups.  - The optimized `find\_duplicates\_optimized` function has an average time complexity of O(n).  Impact of using a set:  - The optimized code uses a set (`seen\_elements`) to store numbers encountered so far.  - Checking if an element is in a set has an average time complexity of O(1), which is constant time regardless of the size of the set.  - In contrast, the original code likely used list lookups (e.g., 'num not in duplicates'), which have a time complexity of O(k), where k is the size of the list being searched. This could lead to significantly slower performance for larger lists.  Impact of single iteration:  - The optimized code iterates through the input list only once.  - The original code used nested loops, meaning for each element in the list, it iterated through the list again. This resulted in O(n^2) operations just for the iterations.  - The single pass in the optimized code directly contributes to its linear time complexity.  Summary of Efficiency Gains:  The optimized code is significantly faster for larger datasets primarily due to:  1. Using a set for near constant-time lookups (O(1) on average) compared to potentially linear-time list lookups (O(k)).  2. Employing a single pass through the input list instead of nested loops, reducing the number of operations from O(n^2) to O(n).  These factors combine to transform the time complexity from roughly O(n^3) to O(n), making the optimized code much more scalable and efficient for large inputs.  **Task 3: Code Refactoring for Readability**  **Task**: Refactor messy code into clean, PEP 8–compliant, well-structured code.  # buggy\_code\_task3.py  def c(n):  x=1  for i in range(1,n+1):  x=x\*i  return x  print(c(5))  Expected Output:  Function renamed to calculate\_factorial.  Proper indentation, variable naming, docstrings, and formatting.  AI should provide a more readable version.  **Prompt:**    **Code:**      **Output:**    **Explanation:**    **Task 4: Security and Error Handling Enhancement**  **Task:** Add security practices and exception handling to the code.  # buggy\_code\_task4.py  import sqlite3  def get\_user\_data(user\_id):  conn = sqlite3.connect("users.db")  cursor = conn.cursor()  query = f"SELECT \* FROM users WHERE id = {user\_id};" # Potential SQL injection risk  cursor.execute(query)  result = cursor.fetchall()  conn.close()  return result  user\_input = input("Enter user ID: ")  print(get\_user\_data(user\_input))  **Expected Output:**  Safe query using parameterized SQL (? placeholders).  Try-except block for database errors.  Input validation before query execution.  **Prompt:**    **Code:**        **Output:**    **Task 5: Automated Code Review Report Generation**  **Task**: Generate a **review report** for this messy code.  # buggy\_code\_task5.py  def calc(x,y,z):  if z=="add":  return x+y  elif z=="sub": return x-y  elif z=="mul":  return x\*y  elif z=="div":  return x/y  else: print("wrong")    print(calc(10,5,"add"))  print(calc(10,0,"div"))  **Expected Output**:  AI-generated **review report** should mention:   * + Missing docstrings   + Inconsistent formatting (indentation, inline return)   + Missing error handling for division by zero   + Non-descriptive function/variable names   + Suggestions for readability and PEP 8 compliance   **Prompt:**    **Code:**    **Output:**    **Explanation:** | | | | | | Week5 - Thursday |  |