|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOLOFCOMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENTOFCOMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **AssignmentType: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | 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) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **DateandDay**  **of Assignment** | | | Week7 - WednesDay | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:13.3**(Presentassignmentnumber)/**24**(Totalnumberofassignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | **Lab 13 – Code Refactoring: Improving Legacy Code with AI Suggestions**  **Lab Objectives**   * To introduce the concept of code refactoring and why it matters (readability, maintainability, performance). * To practice using AI tools for identifying and suggesting improvements in legacy code. * To evaluate the before vs. after versions for clarity, performance, and correctness. * To reinforce responsible AI-assisted coding practices (avoiding over-reliance, validating outputs).   **Learning Outcomes**  After completing this lab, students will be able to:   1. Use AI to analyze and refactor poorly written Python code. 2. Improve code **readability, efficiency, and error handling**. 3. Document AI-suggested improvements through comments and explanations. 4. Apply refactoring strategies without changing functionality. 5. Critically reflect on AI’s refactoring suggestions.   **Task Description #1 – Remove Repetition**  Task: Provide AI with the following redundant code and ask it to refactor  **Python Code**  def calculate\_area(shape, x, y=0):  if shape == "rectangle":  return x \* y  elif shape == "square":  return x \* x  elif shape == "circle":  return 3.14 \* x \* x    **Expected Output**   * Refactored version with dictionary-based dispatch or separate functions. * Cleaner and modular design.   PROMPT: Refactor the shape area calculator using dictionary-based dispatch or separate functions for cleaner, modular design. Ensure each shape's logic is isolated and easily extendable.  CODE SCREENSHOT:  **Observation:**  The refactored code using dictionary-based dispatch is clean, modular, and easy to extend. Each shape's logic is separated into its own function, improving readability and reducing repetition. The use of a dispatch table simplifies control flow and makes the program scalable for adding more shapes. Overall, it's a well-structured and efficient design.  **Task Description #2 – Error Handling in Legacy Code**  Task: Legacy function without proper error handling  **Python Code**  def read\_file(filename):  f = open(filename, "r")  data = f.read()  f.close()  return data  **Expected Output:**  AI refactors with with open() and try-except:  PROMPT:  Write a Python program that asks the user for a filename and reads its contents. Use with open() and try-except blocks to handle file errors gracefull  CODE SCREENSHOT:    OBSERVATION:  This Python function read\_file() prompts the user to enter a filename, then attempts to open and read the file using a with open() block. It handles errors gracefully using try-except, specifically catching FileNotFoundError and IOError, and prints appropriate messages if the file cannot be accessed  **Task Description #3 – Complex Refactoring**  Task: Provide this legacy class to AI for readability and modularity improvements:  **Python Code**  class Student:  def \_\_init\_\_(self, n, a, m1, m2, m3):  self.n = n  self.a = a  self.m1 = m1  self.m2 = m2  self.m3 = m3  def details(self):  print("Name:", self.n, "Age:", self.a)  def total(self):  return self.m1+self.m2+self.m3  **Expected Output:**   * AI improves naming (name, age, marks). * Adds docstrings. * Improves print readability. * Possibly uses sum(self.marks) if marks stored in a list.   PROMPT:  Write a Python program using a class to represent a student with name, age, and marks in three subjects. Accept user input to create a student object, display the student's details, and calculate the total marks.  CODE SCREENSHOT:    OBSERVATION:  The code effectively uses a class to represent a student with attributes for name, age, and marks. It follows good object-oriented practices by encapsulating data and behavior. The use of a list for marks and sum() for total calculation makes it scalable and clean. User input integration is smooth, and the docstrings enhance readability. Overall, it's a well-structured and interactive design suitable for basic student data management.  **Task Description #4 – Inefficient Loop Refactoring**  Task: Refactor this inefficient loop with AI help  **Python Code**  nums = [1,2,3,4,5,6,7,8,9,10]  squares = []  for i in nums:  squares.append(i \* i)  **Expected Output:** AI suggested a **list comprehension**  **PROMPT:** Refactor the loop that calculates squares of numbers using list comprehension for improved readability and performance.  **CODE SCREENSHOT:**  **OBSERVATION:**  The refactored code replaces the traditional for loop with a concise list comprehension, improving readability and reducing lines of code. This approach is more Pythonic and efficient for simple transformations. It maintains the same functionality while making the intent of the code clearer. Overall, it's a clean and elegant improvement suitable for modern Python practices. | | | | | | Week5 - Monday |  |