|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:**  ***improve the above python code with readability, efficiency, and error handling and refactoring it with dictionary-based dispatch or separate functions***  **Code & Output:**    **Explanation:**   * **Before: Mixed styles (if/elif plus dispatch), duplicate calculate\_area, dispatch map references functions before they’re defined, handlers called with wrong arity, uses 3.14, missing validation and docstring, and unreachable print statements inside the function.** * **Now: Clean dispatch-based design with area\_\* functions, math.pi, input and shape validation, case-insensitive shapes, and a documented calculate\_area that only passed needed args.**   **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:**  Refactor the code with open() and try-except block and initiate the Legacy function with proper error handling  **Code & Output:**    **Explanation:**   * Before: Opened the file with open(...), read it, and manually closed it. No error handling, so any I/O issue would raise uncaught exceptions; encoding wasn’t specified. * Now: Uses a with open(..., encoding="utf-8") context manager to ensure the file is always closed. Adds specific try/except blocks for FileNotFoundError, PermissionError, UnicodeDecodeError, and OSError, plus a final catch-all. On failure, prints a clear message and returns None.   **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 to improve the code with addingnaming (name, age, marks).and explain with google style docstrings, enhance readability  And maintain marks in a list.  EX:  Input:  Enter student's name: tej  Enter student's age: 17  Enter marks 556  Output:  Name: tej | Age: 17  Marks: [556] | Total: 556  **Code & Output:**    **Explaination:**   * **Renamed attributes and parameters to clear names (name, age, marks) for readability.** * **Added docstrings to the class and methods to describe purpose and behavior.** * **Switched to f-strings for cleaner, more readable output in details().** * **Stored marks in a list and compute totals with sum(self.marks).** * **Made \_\_init\_\_ accept either multiple mark values or a single iterable of marks.** * **Added a user-interactive main block to input name, age, and marks, then print details and total.**   **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 code with list comprehension in it and make it more readable and easy to understand  **Code & Output:**    **Explaination:**   * I replaced the explicit loop and append with a clear list comprehension: squares = [n \* n for n in nums], which is shorter and easier to read. * I formatted the nums list with consistent spacing for readability. * You then added print(squares) to display the resulting list of squared numbers. | | | | | | Week5 - Monday |  |