**NAME:JAITARAM RUTHVIK REDDY BATCH:04 ROLL NO:2403A510B5**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | Venkataramana Veeramsetty | | | | | |
| **Instructor(s)Name** | | | | 1. Dr. Mohammed Ali Shaik  2. Dr. T Sampath Kumar  3. Mr. S Naresh Kumar  4. Dr. V. Rajesh  5. Dr. Brij Kishore  6. Dr Pramoda Patro  7. Dr. Venkataramana  8. Dr. Ravi Chander  9. Dr. Jagjeeth Singh | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week2-Tuesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | | 24CSBTB01 To 24CSBTB39 | | | |
| **AssignmentNumber:3.2**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 3: Prompt Engineering – Improving Prompts and Context Management  **Lab Objectives:**   * To understand how prompt structure and wording influence AI-generated code. * To explore how context (like comments and function names) helps AI generate relevant output. * To evaluate the quality and accuracy of code based on prompt clarity. * To develop effective prompting strategies for AI-assisted programming.   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Generate Python code using Google Gemini in Google Colab. * Analyze the effectiveness of code explanations and suggestions by Gemini. * Set up and use Cursor AI for AI-powered coding assistance. * Evaluate and refactor code using Cursor AI features. * Compare AI tool behavior and code quality across different platforms.   **Task Description#1**   * Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example   **Expected Output#1**   * Comparison of AI-generated code styles   **Prompt:**  🡪Write a Python program that defines a function to calculate compound interest. The program should ask the user to input the principal, annual interest rate, time in years, and the number of times interest is compounded per year. Then calculate and display both the final amount and the compound interest.  **Output:**    **Observation:**  🡪The program defines a reusable function to calculate compound interest using the correct formula. It takes dynamic user inputs (principal, rate, time, compounding frequency) and outputs both the final amount and interest earned. The code is clean, well-documented with a docstring, and user-friendly with formatted results. Improvements could include input validation and handling edge cases.  **Task Description#2**   * Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.   **Expected Output#2**   * AI-generated function evolves from unclear to accurate multi-statistical operation.   **Prompt:**  🡪Write a Python program that defines a function to calculate the average, median, and mode of a list of numbers. The program should accept the list as dynamic input from the user and then display the results clearly.  **Output:**    **Observation:**  🡪 The program calculates the **average, median, and mode** of a list of numbers entered by the user. It uses a function for modularity and Python’s statistics module for accurate computation of median and mode, while the average is calculated manually. The program accepts dynamic input, handles exceptions for mode, and displays the results in a clear and formatted manner.  **Task Description#3**   * Provide multiple examples of input-output to the AI for convert\_to\_binary(num) function. Observe how AI uses few-shot prompting to generalize.   **Expected Output#3**   * Enhanced AI output with clearer prompts   **Prompt:**  🡪Write a Python function to convert a positive integer to binary without using bin(). Take user input and display the result.  **Output:**    **Observation:**  🡪The program converts a positive integer into its binary representation without using the built-in bin() function. It uses repeated division by 2 and concatenation of remainders to form the binary string, and displays the correct result based on user input.  **Task Description#4**   * Create an user interface for an hotel to generate bill based on customer requirements   **Expected Output#4**   * Consistent functions with shared logic   **Prompt:**  **🡪**Write a Python program that takes item names, prices, and quantities as input from the user, applies a user-given GST percentage, and generates a hotel bill showing subtotal, GST, and total.  **Output:**      **Observation:**  🡪The program generates a hotel bill based on user input. It collects item details (name, price, quantity), calculates the subtotal, applies GST entered by the user, and displays a neat receipt showing itemized costs, subtotal, GST, and final total. It demonstrates loops, input handling, and formatted output in Python.    **Task Description#5**   * Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions   **Expected Output#5**   * Code quality difference analysis for various prompts   **Prompt:**  🡪Write a Python function that converts temperatures between Celsius and Fahrenheit. The function should take two parameters: the temperature value and the unit ('C' or 'F'). It should return the converted temperature. Then write code to prompt the user for input, call the function, and display the result with proper labels.  **Output:**    **Observation:**  The task shows that more specific prompts produce better code. Vague prompts yield minimal, less structured code, while detailed prompts result in modular, readable, and user-friendly temperature conversion functions. Clear instructions improve code quality, reusability, and usability.  **Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Task#1 | 0.5 | | Task#2 | 0.5 | | Task #3 | 0.5 | | Task #4 | 0.5 | | Task #5 | 0.5 | | **Total** | **2.5 Marks** | | | | | | | 03.08.2025 EOD |  |