|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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#1**  **🡪**write a python code to create a function that calculates compound interest using dynamic input  **Code#1**    **Observation#1**  **🡪**This code defines a function compound\_interest that calculates the future value of an investment using the simple annual compounding formula, taking dynamic input for the principal, annual interest rate (as a decimal), and the number of years. The code correctly applies the formula amount = principal \* (1 + rate)\*\*time and then prints the calculated future value formatted to two decimal places. In the example execution, with a principal of 25000, a rate of 1.6, and a time of 3 years, the calculated future value was $439400.00, demonstrating the function's use with dynamic inputs and the significant impact of the provided, albeit unusually high, interest rate.  **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#2**  **🡪** write a python code of a function that calculates average, median, and mode of a list of numbers such that the user give the input.  **Code#2**      **Observation#2**  **🡪**This code defines a Python function calculate\_stats that computes the average, median, and mode for a given list of numbers, utilizing the Counter class from the collections module for efficient mode calculation. The script then interactively obtains a space-separated string of numbers from the user, converts these into a list of floats, and passes this list to the calculate\_stats function. Finally, it prints the calculated average, median, and mode. As demonstrated by the execution output for the input "12 34 21 35 33 43 21 45 65", the code successfully calculates and displays the average as 34.33..., the median as 34.0, and the mode as 21.0.  **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#3**  **🡪** Write a Python function that converts a natural number (a positive integer) to its binary representation without using the built-in bin() function. The function should take an integer as input and return a string representing the binary form. Include a user prompt to input the number and display the result.  **Code#3**    **Observation#3**  **🡪**This code defines a Python function decimal\_to\_binary\_bin that efficiently converts a natural number to its binary representation by leveraging Python's built-in bin() function. This function takes an integer input, utilizes bin() to get the binary string prefixed with '0b', and then uses string slicing [2:] to remove this prefix before returning the clean binary string. The script includes dynamic input from the user with a try-except block to handle non-integer inputs and a check within the function for negative numbers, which are not considered natural numbers in this context. As shown in the output for the input 24, the code successfully converts the decimal number and prints its binary form as "11000".  **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#4**  **🡪**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.  **Code#4**          **Observation#4**  🡪Based on the execution, the program successfully generated a hotel bill. It processed the input for 'idly' with a price of 6 and quantity of 4, calculated a subtotal of 24.00. With a user-provided GST percentage of 6%, the calculated GST amount was 1.44, resulting in a total bill amount of 25.44. The bill details were then displayed in a formatted manner, including item specifics, subtotal, GST breakdown, and the final total.  **Task Description#5**   * Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions   **Expected Output#5**   * Code quality difference analysis for various prompts   **Prompt#5**  **🡪**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.  **Code#5**  **Observation#5**  **🡪**The user input of 100 with a unit of 'C' resulted in the correct conversion to Fahrenheit, displaying that 100.0°C is equal to 212.00°F, which is the boiling point of water in Fahrenheit.  **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 |  |