|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.N.** | **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**  1.Write a Python function named calculate\_compound\_interest  2.Add a docstring to the function  3.Add an example in the docstring showing input and output  **Code #1:**  IMG_256  **Observations #1:**   1. Final amount means money you get back (your money + interest). 2. Compound interest means only the extra money you earn. 3. With input, the program asks the user directly, so it works for any values.   **Code Explanation #1:**   1. We **define a function** calculate\_compound\_interest to do the math. 2. Inside the function, we use the formula. 3. Then we **subtract the principal** to get only the interest. 4. The program **asks the user** to enter principal, rate, and time. 5. It **calls the function** with those values. 6. Finally, it **prints the interest earned**.   **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:**   1. Takes a list of numbers as input 2. Find average 3. Find median 4. Find mode 5. Returns or prints them   **Code #2:**  IMG_256  **Observations #2**   1. average tells the overall value (add all numbers ÷ count). 2. median is the middle value when numbers are sorted. 3. mode is the number that comes most often. 4. All three give different ways to understand the data. 5. Useful in maths, statistics, and real-life data analysis.   **Code Explanation #2:**  The program asks numbers → checks input → finds average, median, mode → prints them nicely.  **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:**   1. Write a Python function named convert\_to\_binary(num) 2. Now write the function   **Code #3:**  IMG_256  **Observations #3:**   1. Giving **examples of input and output** helps AI understand the pattern. 2. AI looks at the examples and learns how to convert any number into binary. 3. Example: If it knows 5 → 101 and 10 → 1010, it can guess 7 → 111. 4. More examples = better generalization.   **Code Explanation #3**   1. **Ask the user** to enter a number (input). 2. **Convert** the input into an integer (int). 3. If the number is **0**, it directly prints "0". 4. If the number is **negative**, it shows a message (this function does not handle negatives). 5. When the number becomes 0, print the full binary string. 6. If the user enters something that is not a number, it prints "Invalid 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 #4**  1. Ask the customer for room type (Single, Double, Suite)  2. Ask how many nights they will stay  3. Ask if they want extra services (like food, laundry, etc.)  4. Calculate the total bill  5. Print the final bill clearly  **Code#4**  IMG_256  IMG_256  **Observations #4**   1. The program asks the customer for their choices (room type, nights, services). 2. Each choice adds to the total bill. 3. The calculation is simple: **room price × nights + extra charges**. 4. The bill is displayed clearly at the end. 5. Such programs make hotel billing easier and faster.   **Code Explanation #4**   1. It calculates the **total hotel bill** based on:   >Room type  >Number of nights  >Extra services (like food, laundry, other)  **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**  1.Write a Python function to convert Celsius to Fahrenheit  2.Now improve the function to also convert Fahrenheit to Celsius  3.Make the program ask the user for input and print the result  **Code #5**  IMG_256  **Observations #5**   1. Program lets user choose conversion direction (C → F or F → C). 2. It asks for a temperature value and checks for valid number input. 3. If user enters wrong input, it asks again until correct. 4. Uses correct formulas for both conversions. 5. Prints the result in a clear sentence   **Code Explanation#5**   1. while True loops are used to keep asking until input is valid. 2. direction = input(...).upper() takes user choice and converts to uppercase.·   >"C" = Celsius → Fahrenheit  >"F" = Fahrenheit → Celsius   1. temperature = float(input(...)) ensures the input is a number. 2. If **C → F**: 3. > °F=(°C×9/5)+32°F = (°C \times 9/5) + 32°F=(°C×9/5)+32 4. If **F → C**:   >°C=(°F−32)×5/9°C = (°F - 32) \times 5/9°C=(°F−32)×5/9  7. Finally, program prints the result with one decimal place.    **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 |  |