**Name: Gundu Meghana**

**Enrollment No: 2403A510C1**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:** Develop a Python function named calculate\_compound\_interest(). Start by creating just the function signature. Then add a detailed docstring explaining the purpose and parameters. Follow this by implementing the compound interest calculation logic. Finally, provide an example showing how to call this function with sample inputs.    **Observation:**   * This task demonstrates a stepwise approach to building a Python function for calculating compound interest. It starts with just the function name, then adds a docstring for clarity, followed by implementation and example usage. The progression highlights how prompt refinement helps generate clearer, better-structured code and facilitates comparison of different code styles.   **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 function that takes a list of numbers as input and returns three statistical values: average, median, and mode. Make sure the function handles cases where the mode might not be unique or present. Use Python’s statistics module where appropriate.       **Observation:**   * The task evolves from performing simple math operations to creating a comprehensive function that calculates average, median, and mode for a list of numbers. It shows the importance of refining prompts to achieve accurate statistical computation and includes handling edge cases such as multiple or no unique modes, improving the robustness of  AI-generated code.   **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.   **Output#3**   * Enhanced AI output with clearer prompts   **Prompt:**   * Using few-shot prompting, define a function convert\_to\_binary(num) that converts an integer to its binary string representation without the "0b" prefix.     **Observation:**   * By giving multiple input-output examples for the binary conversion function, this task demonstrates how few-shot prompting enhances the AI’s ability to generalize the conversion logic effectively. It results in improved clarity and correctness of the function, illustrating the power of contextual examples in prompt engineering.   **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:**   * Create a Python-based user interface for a hotel billing system. The interface should accept customer details like name, room type, number of nights, and additional services (e.g., breakfast, Wi-Fi, laundry). On submission, it should calculate and display the total bill based on the items selected.       **Observation:**   * By giving multiple input-output examples for the binary conversion function, this task demonstrates how few-shot prompting enhances the AI’s ability to generalize the conversion logic effectively. It results in improved clarity and correctness of the function, illustrating the power of contextual examples in prompt engineering.   **Task Description#5**   * Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions   **Expected Output#5**   * Code quality difference analysis for various prompts   **Prompt:**   * Construct a Python function named convert\_temperature that performs temperature conversion between Celsius (°C) and Fahrenheit (°F).     **Observation:**   * Focused on prompt specificity, this task uses clear instructions to improve a temperature conversion function. It requires input validation, meaningful variable naming, and demonstration of usage with examples. This results in higher-quality, error-resistant code and illustrates how detailed prompts elevate AI output in terms of usability and correctness.   **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 |  |