|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | |  | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:3.3**(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**   * Try 3 different prompts to generate a factorial function.   **Expected Output#1**   * Comparison of AI-generated code styles   Prompt 1:python code to generate a factorial of a number and number should be taken from user  Code:    **Output:**    **Prompt2:python code for factorial of a number 10**  **Code:**    **Output:**    **Prompt3:python code for factorial of a number don’t take number from console**  **Code:**  **Output:**  **Conclusion:**  **All three codes calculate factorial using the same logic, but they differ in how the input is given. task1.py takes input from the user at runtime, making it interactive. task1\_2.py uses a hardcoded value 10, so it always calculates factorial of 10. task1\_3.py also uses a hardcoded value 5, but adds a helpful comment suggesting the value can be changed. In short, the difference lies only in input method—user input vs fixed value.**  **Task Description#2**   * Provide a clear example input-output prompt to generate a sorting function.   **Expected Output#2**   * Functional sorting code from AI   Prompt 1: Provide a clear example input-output prompt to generate a sorting function.  Code:    Output:    **Prompt2:Python code for sorting by taking input from console**  **Code:**    **Output:**    **Prompt3:python code without inbuilt function for sorting**  **Code:**  **Output:**    **Conclusion:**  **All three codes sort a list but differ in approach. task2\_1.py and task2\_2.py use Python’s inbuilt sorted() function, making them short, efficient, and practical for real-world use. task2\_3.py implements sorting manually using the Bubble Sort algorithm, which is longer and slower.**  **Task Description#3**   * Start with the vague prompt “Generate python code to calculate power bill” and improve it step-by-step   **Expected Output#3**   * Enhanced AI output with clearer prompts   **Prompt1:python Code to calculate power bill based on number of customers and type of customers**  **Code:**      **Output:**    **Prompt2: python Code to calculate power bill based on number of customers and type of customers ask all types of costs from console**  **Code:**      **Output:**    **Prompt3: : python Code to calculate power bill based on number of customers and type of customers ask all types of costs from console and per unit cost should be different for different range of consumption**  **Code:**      **Output:**    **Conclusion:**  **These three codes all calculate electricity bills but differ in flexibility. task3\_1.py uses fixed rates (Residential = 5, Commercial = 8, Industrial = 10) for billing, making it the simplest. task3\_2.py improves flexibility by letting the user enter custom per-unit rates for each type before billing. task3\_3.py is the most advanced, as it allows defining multiple slabs (consumption ranges with different rates) for each customer type, making it closer to real-world billing systems. In short, the difference is in rate handling—fixed, user-defined, and slab-based.**  **Task Description#4**   * Write structured comments to help AI generate two linked functions (e.g., login\_user() and register\_user()).   **Expected Output#4**   * Consistent functions with shared logic   **Prompt1:** Write python code to help AI generate two linked functions (e.g., login\_user() and register\_user()).  **Code:**      **Output:**    **Prompt2:** Write python code to help AI generate two linked functions (e.g., login\_user() and register\_user()). Don’t exit without making login  **Code:**      **Output:**  **Conclusion:**  **The codes show two different implementations of a user registration and login system in Python. The key difference lies in their main logic loop and the user's ability to exit.**  **The first version is designed to force the user to register and log in before they can proceed. The while loop continues as long as logged\_in is False. There is no option to exit the program without successfully logging in. The loop terminates only after a successful login, at which point it prints "You are now logged in. Proceed with your tasks."**  **The second version gives the user more control by providing an exit option. The while True loop runs indefinitely until the user explicitly chooses to exit by selecting option "3". The break statement then terminates the loop and the program. This version doesn't force a login; it allows the user to register, log in, or exit at their discretion.**  **Task Description#5**   * Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions   **Expected Output#5**   * Code quality difference analysis for various prompts   **Prompt1:Python code for temperature conversion from Fahrenheit to celsius**  **Code:**    **Output:**    **Prompt2:python code for temperature conversion by taking temperature from console**  **Code:**    **Output:**    Conclusion:  The two Python code snippets both deal with temperature conversion, but they differ in their **structure and modularity**.  The first code uses a **single function, temperature\_conversion()**, to handle both Celsius to Fahrenheit and Fahrenheit to Celsius conversions. It prompts the user for their choice of conversion type within this function and then performs the calculation and prints the result directly.  In contrast, the second code demonstrates a **more modular approach**. It defines a **dedicated function, fahrenheit\_to\_celcius()**, specifically for converting Fahrenheit to Celsius. This function takes a Fahrenheit temperature as input and *returns* the calculated Celsius value, separating the conversion logic from the user interaction and output. If a Celsius to Fahrenheit conversion was also needed, a separate function for that would likely be created, promoting code reusability and better organization.  **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** | | --- | --- | | Factorial Function (Task#1) | 0.5 | | Sorting Function (Task#2) | 0.5 | | Vogue Vs. Specific Prompting (Task #3) | 0.5 | | Linked Functions (Task #4) | 0.5 | | Temperature Conversion Function (Task #5) | 0.5 | | **Total** | **2.5 Marks** | | | | | | | 03.08.2025 EOD |  |