|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 : write a python function to calculate factorial**      **Prompt 2: Write python function that take input from user and return factorial of the number user entered**      **Prompt 3 : Create a Python program to compute the factorial of a number using recursion with proper function docstring.**      **CONCLUSION : These three codes all do the same job—finding the factorial of a number—but in different ways. The first one is simple and straight to the point, good if you just want to use it inside another program. The second one feels more user-friendly since it asks for input and handles errors, but it’s less flexible for reuse. The third one uses recursion, which looks neat and matches the mathematical definition, but it’s not the best for very large numbers. In short, each code has the same purpose but fits different situations depending on what you need.**  **Task Description#2**   * Provide a clear example input-output prompt to generate a sorting function.   **Expected Output#2**   * Functional sorting code from AI   **Prompt : Write a Python function that takes a list of integers as input and returns the list sorted in descending order**      **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   **Prompt : Write a Python program that takes the number of electricity units consumed as input and calculates the total bill based on the following customer**      **Prompt 2 : Generate Python code to calculate an electricity bill based on the number of units consumed, using a fixed rate per unit.**      **Comclusion : Both codes calculate an electricity bill but in different styles. The first one is very simple—you just give it the units and rate, and it multiplies them. The second one is more realistic because it uses slab rates, like how actual electricity bills are calculated, and also checks for errors. In short, the first is basic and flexible, while the second feels more practical and closer to real life.**  **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   **Prompt : generate a two linked functions in python (Celsius,fahrenheit)**      **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 to convert Celsius to Fahrenheit**      **Prompt : Write a Python function that converts temperature between Celsius and Fahrenheit, based on user input.**      **Conclusion : The first code is simple and reusable for Celsius to Fahrenheit, while the second is interactive and lets users convert both ways, making it more versatile.**  **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 |  |