|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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   Prompt1: write a python function generate factorial    Output:    Prompt2:  write a python function generate factorial that value given by user    Output:    Prompt3: write a python function that generate factorial up to n number .    Output:    **Explanation :** task1\_1: Computes a single n! and prints demo outputs.  task1\_2: Computes a single n! from user input, validates non-integer input.  task1\_3: Returns a list of factorials from 0! to n!.  **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 code that generate sorting function that value given by user.    Output:    **Explanation:**  Splits the input string into a list of tokens.  Tries to convert all tokens to integers; if successful, sorts them numerically.  If any token isn’t an integer, it sorts all tokens as strings (alphabetically).  Prints the sorted list and calls the function to run immediately.    **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 code that generate the calculate of a power bill.    Output:    Prompt : write a python code that generate the calculate power bill and says the price of the things in pill with gst.    **Output:**    **Explanation:**  **Both scripts define calculate\_power\_bill() that asks for the number of items and validates it.**  **For each item, they take name and price, validate the price, and compute GST at 18%.**  **They store each item’s original and GST-added price, then print detailed line items.**  **Difference: Only the docstring wording differs (task3\_1.py mentions appliances; task3\_2.py uses more generic phrasing). Logic and output are identical**  **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 : write a python code that generate two linked function(login\_user() and register\_user()**    **Output:**    **Explanation:**  **register\_user() saves a new username/password in users\_db if not taken; login\_user() checks credentials, no persistence or hashing.**  **Task Description#5**   * Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions   **Expected Output#5**   * Code quality difference analysis for various prompts   Prompt1: write a python code that improving temperature conversion function with clear instruction.        Output:    **Prompt2:** write a python code improving temperature conversion function with clear instruction  With whether condition .        **Output:**  **.**  **Explanation:**  **task5\_1: Uses many pairwise if/elif conversions; simple result only.**  **task5\_2: Converts via Celsius (cleaner/extensible), clearer prompts, and adds weather advice.**  **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 |  |