|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | Venkataramana Veeramsetty | | | | | |
| **Instructor(s) Name** | | | |  | | | | | |
| **Course Code** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week2 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **Assignment Number:4.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques    **Lab Objectives:**     * To explore and apply different levels of prompt examples in AI-assisted code generation. * To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality. * To evaluate the impact of context richness and example quantity on AI performance. * To build awareness of prompt strategy effectiveness for different problem types.   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:     * Use zero-shot prompting to instruct AI with minimal context. * Use one-shot prompting with a single example to guide AI code generation. * Apply few-shot prompting using multiple examples to improve AI responses. * Compare AI outputs across the three prompting strategies.         **Task #1 – Zero-Shot Prompting with Conditional Validation**  Objective  Use zero-shot prompting to instruct an AI tool to generate a function that validates an Indian mobile number.  Requirements   * The function must ensure the mobile number:   + Starts with 6, 7, 8, or 9   + Contains exactly 10 digits   Expected Output   * A valid Python function that performs all required validations without using any input-output examples in the prompt.     006eacb6bd80fe9d8460f38476cd4a7f.png    **OBSERVATIONS:**  **The function validate\_mobile correctly implements all required checks for an Indian mobile number:**   1. **Ensures the input is a string.** 2. **Checks the length is exactly 10 characters.** 3. **Verifies that all characters are digits.** 4. **Validates the first digit is 6, 7, 8, or 9.**   **The while True loop then repeatedly takes user input and validates it until an invalid number is entered, at which point it displays an error message and stops execution.**  **This approach ensures continuous validation without relying on pre-given examples, but it will terminate immediately on the first invalid input.**  **-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**  **Task #2 – One-Shot Prompting with Edge Case Handling**  Objective  Use one-shot prompting to generate a Python function that calculates the factorial of a number.  Requirements   * Provide one sample input-output pair in the prompt to guide the AI. * The function should handle:   + 0! correctly   + Negative input by returning an appropriate message   Expected Output   * A Python function with correct factorial logic and edge case handling, generated from a single example.   cd000ab56935da5eca7b14a878c19976.png    **OBSERVATIONS:**  The generated factorial function:   * Returns False if the input is a negative number. * Raises a TypeError if the input is not an integer. * Returns 1 when the input is zero (0! = 1). * Calculates the factorial iteratively for positive integers, avoiding recursion depth limitations.     **------------------------------------------------------------------------------------------------------------**      **Task #3 – Few-Shot Prompting for Nested Dictionary Extraction**  Objective  Use few-shot prompting (2–3 examples) to instruct the AI to create a function that parses a nested dictionary representing student information.  Requirements   * The function should extract and return:   + Full Name   + Branch   + SGPA   Expected Output   * A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples.   a6291011469b0435e5d6cf5d447a4331.png    OBSERVATIONS:  Few-shot prompting made the AI easily infer the **nested access pattern** without extra clarification. It’s reusable and works for any similar dictionary.  -----------------------------------------------------------------------------------------------------------------------------------------------------------  **Task #4 – Comparing Prompting Styles for File Analysis**  Objective  Experiment with zero-shot, one-shot, and few-shot prompting to generate functions for CSV file analysis.  Requirements   * Each generated function should:   + Read a .csv file   + Return the total number of rows   + Count the number of empty rows   + Count the number of words across the file   Expected Output   * Working Python functions for each prompting style, with a brief reflection comparing their accuracy, clarity, and efficiency.   16af4cc9e165efd48cd414e2a555db1c.png    6f28d11daaa7b33a581d459c7264e19b.png  8df003840a269d7e0d1727ff3bd2c284.png  **OBSERVATIONS:**  **Observation:**   * **Zero-shot** worked, but sometimes miscounted empty rows. * **One-shot** improved accuracy by showing one example. * **Few-shot** was most reliable and consistent for complex CSV structures.     **Task #5 – Few-Shot Prompting for Text Processing and Word** **Frequency**  Objective  Use few-shot prompting (with at least 3 examples) to generate a Python function that processes text and analyzes word frequency.  Requirements  The function must:   * Accept a paragraph as input * Convert all text to lowercase * Remove punctuation * Return the most frequently used word   Expected Output   * A functional Python script that performs text cleaning, tokenization, and returns the most common word using only the examples provided in the prompt   **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  fd6a680bf5679547776a6911eafbcfcc.png  OBSERVATIONS:  Few-shot prompting gave a clear pattern — convert to lowercase, strip punctuation, then count words. It avoided edge case errors like punctuation splitting.      **Evaluation Criteria:** | | | | | | Week2 - Monday |  |