**Course Code: AIPP**

**Assignment No: 4**

**Done by: 2503B09905 (MCA)**

**Name: KVN Jayanth**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**M.Tech. and MCA | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | Venkataramana Veeramsetty | | | | | |
| **CourseCode** | | |  | **CourseTitle** | | AI Assisted Problem Solving Using Python | | | |
| **Year/Sem** | | | I/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week3 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | | M.Tech. and MCA | | | |
| **AssignmentNumber:4.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***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 Description#1**   * Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.   **Expected Output#1**   * AI-generated function with no examples provide.   **Prompt:**   * Write a Python function to check if a given year is a leap year.   **Generated Code:**    **Explanation**   * **The condition checks leap year logic:**   + Divisible by 4 → leap year   + Except divisible by 100 → not leap year   + Unless divisible by 400 → leap year again * **Example:**   + 2000 ✅ (leap year)   + 1900 ❌ (not leap year)   + 2024 ✅ (leap year)   **Example Output:**    **Task Description#2**   * One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.   **Expected Output#2**   * Function with correct conversion logic   **Prompt:**   * Write a Python function to convert centimeters to inches. Example: 10 cm = 3.937 inches.   **Generated Code:**      **Explanation:**   * The function converts centimeters to inches using the formula **cm / 2.54**. * It uses **type hints** and a **docstring** for clarity, and raises a **ValueError** if a negative value is given. * The example block tests a few values and prints neatly formatted results.   **Example Output:**    **Task Description#3**   * Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.   **Expected Output#3**   * Well-structured function respecting the examples   **Prompt:**   * Generate a function that formats full names as “Last, First.   **Examples:**   * Jayanth kasarla→ kasarla, Jayanth * Kiran gundam → gundam, Kiran * Karthik darla → darla, Karthik   **Generated Code:**    **Explanation:**   * Function name format\_name() splits the given name into words using split(). * The last word becomes the surname, and the rest are treated as first name(s). * Finally, it joins them in the format “Last, First”. * Few-shot prompting (multiple examples) helped Copilot understand the pattern correctly.   **Example Output:**    **Task Description#4**   * Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.   **Expected Output#4**   * Functional output and comparative reflection   **Prompt:**   * Zero-shot: Write a Python function to count vowels in a string. * Few-shot: Examples — * “hello” → 2 * “AI” → 2 * “Ramesh” → 2   **Generated Code:**    **Explanation**   * **Zero-shot:** Copilot generated a basic version using a string of vowels — correct but slightly less efficient. * **Few-shot:** After giving examples, Copilot used a **set** of vowels for faster membership checking (better performance). * Both give the same results, but few-shot produces more optimized logic.   **Comparative Analysis:**   1. **Implementation Differences**:    * Zero-shot: Simpler, uses .lower(), only checks lowercase vowels    * Few-shot: More robust, handles both cases, uses set for efficiency 2. **Advantages**:    * Zero-shot: More straightforward, less code    * Few-shot: More precise, handles edge cases better   **Example Output:**  **C:\Users\JAYANTH\OneDrive\Pictures\Screenshots\Screenshot 2025-11-08 101028.png**  **Output Comparison:**   * The few-shot approach tends to be more reliable as it's guided by specific examples and edge cases.   **Task Description#5**   * Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.   **Expected Output#5**   * Working file-processing function with AI-guided logic   **Prompt**   * Write a Python function that reads a text file and returns the number of lines. * Use examples like:   + data.txt → 5 lines   + notes.txt → 3 lines   + sample.txt → 10 lines * Include user input to specify the filename and print the result.   **Generated Code:**    **Explanation**   * This function uses with open() to safely open the file in read mode. * It reads all lines using readlines() and returns their count using len(lines). * Few-shot prompting guided Copilot to handle file reading and counting automatically with examples.   **Example Output:**  **C:\Users\JAYANTH\OneDrive\Pictures\Screenshots\Screenshot 2025-11-08 095416.png**  A working Python function that correctly reads a .txt file and returns the total number of lines. **The program should:**   * Accept the filename from the user. * Open and read the file safely. * Count and display the number of lines present in the file.   **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** | | --- | --- | | Zero Shot (Task #1) | 2.5 | | One Shot (Task#2) | 2.5 | | Few Shot (Task#3 & Task #5) | 2.5 | | Comparison (Task#4) | 2.5 | | **Total** | **10 Marks** | | | | | | | Week3 - Monday |  |