|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** M.Tech. and MCA | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **Course Coordinator Name** | | | | Venkataramana Veeramsetty | | | | | |
| **Course Code** | | |  | **Course Title** | | AI Assisted Problem Solving Using Python | | | |
| **Year/Sem** | | | I/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week3 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | M.Tech. and MCA | | | |
| **AssignmentNumber:4.3**(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 Description#1**   * CZero-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 provided       **Task Description#2**   * One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.   Code:  **Expected output#2**  **Task Description#3**   * Few short :provided 2-3 examples to generate a function that formates full names as “Last, Fast”   **Expected Output#3**   * Well-structured function respecting the examples     **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.     **Comparative Reflection**  **Task Description#5**  **●Use few-shot prompting to generate a function that reads a .txt file and returns the Comparative Reflection**  **number of line**   | **Aspect** | **Zero-shot Prompt** | **Few-shot Prompt** | | --- | --- | --- | | **Input Style** | Simple instruction only | Instruction + examples | | **Function Output** | Works correctly, but style varies | Works correctly and follows example style | | **Naming Convention** | Random (count\_vowels) | Consistent with examples (vowel\_counter) | | **Logic** | Explicit loop with counter | Uses concise list comprehension | | **Control over Output** | Limited | High (can guide style, naming, structure) | | **Error Likelihood** | Slightly higher for complex tasks | Reduced (model imitates examples) | | **Best Use Case** | Quick general solutions | Consistent code generation across tasks |   .  **Expected Output#5**   * Working file-processing function with AI-guided logic   **OUTPUT:**  **# Example 1: Function to count words in a .txt file**  **def count\_words\_in\_file(filename):**  **with open(filename, 'r') as file:**  **text = file.read()**  **words = text.split()**  **return len(words)**  **# Example 2: Function to count characters in a .txt file**  **def count\_characters\_in\_file(filename):**  **with open(filename, 'r') as file:**  **text = file.read()**  **return len(text)**  **# AI-guided logic:**  **# Observing both examples, each reads the file, processes the content differently,**  **# and returns a count. So, to count lines, we can follow the same structure but use readlines().**  **def count\_lines\_in\_file(filename):**  **"""Reads a .txt file and returns the number of lines."""**  **try:**  **with open(filename, 'r') as file:**  **lines = file.readlines()**  **return len(lines)**  **except FileNotFoundError:**  **return "Error: File not found."**  **except Exception as e:**  **return f"An error occurred: {e}"**  **# Example usage:**  **# print(count\_lines\_in\_file("sample.txt"))**  **AI-Guided Logic Steps**   1. **Pattern Recognition:** The AI recognized a pattern from examples — open file → process → return count. 2. **Task Adaptation:** Instead of counting words or characters, the AI applies the logic to **lines** using readlines(). 3. **Error Handling:** Added try-except blocks for real-world reliability (e.g., missing files   **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 |  |