|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | |  | | --- | | Dr. V. Venkataramana (Co-Ordinator) | | Dr. T. Sampath Kumar | | Dr. Pramoda Patro | | Dr. Brij Kishor Tiwari | | Dr.J.Ravichander | | Dr. Mohammand Ali Shaik | | Dr. Anirodh Kumar | | Mr. S.Naresh Kumar | | Dr. RAJESH VELPULA | | Mr. Kundhan Kumar | | Ms. Ch.Rajitha | | Mr. M Prakash | | Mr. B.Raju | | Intern 1 (Dharma teja) | | Intern 2 (Sai Prasad) | | Intern 3 (Sowmya) | | NS\_2 ( Mounika) | | | | | | |
| **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 * Prompt: Generate a function in Python that validates an Indian mobile number. The function must check that the number starts with 6, 7, 8, or 9 and contains exactly 10 digits." for the number given by the user     Expected Output   * A valid Python function that performs all required validations without using any input-output examples in the prompt. * Output:     **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   Prompt: Write a python program using functions that calculates the factorial of a number for positive integer only if negative return an appropriate message.  Example: 0! = 0    Expected Output   * A Python function with correct factorial logic and edge case handling, generated from a single example.   **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   Prompt: write a python function that generated the parases a nested dictionary representing student information with user input.  Eg 1: Full name = HARSHA  Branch: CSE  SGPA: 9.5  Eg2: Full name: SUPPU  Branch: EEE  SGPA: 9.4    Expected Output   * A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples.     **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   Prompt1: *Write a Python function that reads a CSV file and returns the total number of rows, the number of empty rows, and the total number of words in the file.*    Prompt2: Here’s an example of a function that reads a CSV and returns the number of rows. Now write one that also counts empty rows and total words.  Example:  def count\_rows(file\_path):  with open(file\_path, newline='', encoding='utf-8') as f:  return sum(1 for \_ in csv.reader(f))      Prompt3: *Here are two examples of functions that read CSV files and perform analysis. Now write one that returns total rows, empty rows, and word count.*  Examples Given:   1. Count rows and columns 2. Count non-empty cells     **Zero-shot** works well for simple tasks but may lack polish or consistency.   * **One-shot** benefits from context and tends to produce cleaner logic.   **Few-shot** shines when clarity and structure matter—especially for maintainable code  **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   Prompt: . Generate a Python function that processes text and finds the most frequent word.The function should:   * Accept a paragraph as input * Convert text to lowercase * Remove punctuation * Return the most frequently used word   **Examples**  **Example 1:** Input: "Hello hello world!" Output: "hello"  **Example 2:** Input: "Python is great, and Python is easy." Output: "python"  **Example 3:** Input: "I love apples, you love apples, we all love apples!" Output: "apples"  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   **Prompt: Write a Python function that takes a paragraph of text and finds the most common word. The function should:**   * **Turn all the text into lowercase** * **Remove any punctuation (like commas, periods, question marks, etc.)** * **Split the text into words** * **Count how often each word appears** * **Return the word that appears the most**   **Example:**  **def to\_lowercase(text):**  **return text.lower()**  **Example:**  **import string**  **def remove\_punctuation(text):**  **return text.translate(str.maketrans('', '', string.punctuation))**  **Example:** **from collections import Counter**  **def count\_words(text):**  **words = text.split()**  **return Counter(words)**    **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) | 0.5 | | One Shot (Task#2) | 0.5 | | Few Shot (Task#3, Task#4 & Task #5) | 1.5 | | **Total** | **2.5 Marks** | | | | | | | Week2 - Monday |  |