|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 python function must ensure the Indian mobile number that starts with 6,7,8 or 9 and contains exactly 10 numbers.  Expected Output     * A valid Python function that performs all required validations without using any input-output examples in the prompt.     **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 : generate a python function that calculate the factorial of a number that handle 0! Correctly and Negative input by returning an appropriate message, if input is -3 the output should be Not defined.     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 : Create a Python function that parses a nested dictionary representing student information. The function should extract and return the full name, branch, and SGPA of each student. * Examples: * Example 1: * Input: * students = { * "student1": {"full\_name": "Alice Smith", "branch": "CSE", "SGPA": 8.5, "age": 20}, * "student2": {"full\_name": "Bob Johnson", "branch": "ECE", "SGPA": 7.8, "age": 21}, * } * Output: * [ * {"full\_name": "Alice Smith", "branch": "CSE", "SGPA": 8.5}, * {"full\_name": "Bob Johnson", "branch": "ECE", "SGPA": 7.8} * ] * Example 2: * Input: * students = { * "student1": {"full\_name": "John Doe", "branch": "ME", "SGPA": 9.0}, * "student2": {"full\_name": "Jane Roe", "branch": "CSE", "SGPA": 8.7}, * "student3": {"full\_name": "Mike Lee", "branch": "EE", "SGPA": 8.0}, * } * Output: * [ * {"full\_name": "John Doe", "branch": "ME", "SGPA": 9.0}, * {"full\_name": "Jane Roe", "branch": "CSE", "SGPA": 8.7}, * {"full\_name": "Mike Lee", "branch": "EE", "SGPA": 8.0} * ] * Task: * Write a function extract\_student\_info(students) that takes a nested dictionary of students as input and returns a list of dictionaries with only "full\_name", "branch", and "SGPA" for each student.   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   ZERO-SHOT PROMPT : Generate a Python function that reads a .csv file and returns the total number of rows, counts the number of empty rows, and counts the number of words across the entire file    OUTPUT :    ONE-SHOT PROMPT : Example:  import csv  def analyze\_csv(filename):  with open(filename, 'r') as file:  rows = list(csv.reader(file))  total\_rows = len(rows)  empty\_rows = sum(1 for row in rows if not any(row))  word\_count = sum(len(" ".join(row).split()) for row in rows)  return total\_rows, empty\_rows, word\_count  Now generate a Python function that reads a .csv file and returns the total number of rows, counts the number of empty rows, and counts the number of words across the entire file.    OUTPUT :    FEW-SHOT PROMPT : Example 1:  import csv  def csv\_summary(filepath):  with open(filepath, newline='') as f:  reader = list(csv.reader(f))  total = len(reader)  empties = sum(1 for row in reader if all(cell.strip()=="" for cell in row))  words = sum(len(" ".join(row).split()) for row in reader)  return total, empties, words  Example 2:  import pandas as pd  def csv\_info(path):  df = pd.read\_csv(path)  total = len(df)  empties = df.isnull().all(axis=1).sum()  words = df.astype(str).apply(lambda x: " ".join(x), axis=1).str.split().str.len().sum()  return total, empties, words  Now generate another Python function that reads a .csv file and returns:   * Total number of rows * Number of empty rows * Total number of words across the entire file     OUTPUT :    Expected Output   * Working Python functions for each prompting style, with a brief reflection comparing their accuracy, clarity, and efficiency.   **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 :  **Example 1:**  import string  from collections import Counter  def most\_common\_word(text):  text = text.lower()  text = text.translate(str.maketrans('', '', string.punctuation))  words = text.split()  counter = Counter(words)  return counter.most\_common(1)[0][0]  **Example 2:**  import re  from collections import Counter  def frequent\_word(paragraph):  paragraph = paragraph.lower()  paragraph = re.sub(r'[^\w\s]', '', paragraph)  words = paragraph.split()  return Counter(words).most\_common(1)[0][0]  **Example 3:**  import string  from collections import Counter  def top\_word(text\_input):  text\_input = text\_input.lower()  text\_input = text\_input.translate(str.maketrans('', '', string.punctuation))  word\_list = text\_input.split()  counter = Counter(word\_list)  top = counter.most\_common(1)  return top[0][0]  **Task:** Generate a new Python function that:   * Accepts a paragraph as input * Converts all text to lowercase * Removes punctuation * Returns the most frequently used word   PROMPT : You are a Python coding assistant. I need a function that takes a paragraph of text, processes it, and returns the most frequently used word. The steps should include converting all text to lowercase and removing punctuation.  Here are some examples:  Example 1:  Input: "Hello world! Hello again."  Output: "hello"  Example 2:  Input: "Testing, testing, one two three. One, two!"  Output: "one"  Example 3:  Input: "Data science is fun. Science is powerful. Data is useful."  Output: "is"  Now write a Python function named most\_frequent\_word that:  - Accepts a paragraph as input  - Converts it to lowercase  - Removes punctuation  - Splits the text into words  - Counts word frequencies  - Returns the most frequent word    OUTPUT :     * 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  **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 |  |