|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | 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) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week2 - Wednesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:2.3**(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 provided   Certainly! The core logic of the is\_leap function follows the rules for determining a leap year:  **Divisible by 4:** A year is generally a leap year if it is divisible by 4. (year % 4 == 0)  **Exceptions for centuries:** However, if a year is divisible by 100 (a century year), it is not a leap year unless... (year % 100 != 0)  **Exception to the exception:** ...it is also divisible by 400. In that case, it is a leap year. (year % 400 == 0)  The if statement combines these conditions:   * (year % 4 == 0 and year % 100 != 0) checks for years divisible by 4 but not by 100 (like 2024). * (year % 400 == 0) checks for century years divisible by 400 (like 2000).   If either of these conditions is true, the function returns True, indicating it's a leap year. Otherwise, it returns False  def is\_leap(year):      """      Checks if a given year is a leap year.      Args:          year: The year to check.  Returns:          True if the year is a leap year, False otherwise.      """      if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):          return True      else:          return False  is\_leap(2007)  Output: False  **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   def cm\_to\_inches(cm):    """Converts centimeters to inches."""    return cm \* 0.393701  cm\_to\_inches(890)  Output: 350.39389  **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   def format\_name(full\_name):    """    Formats a full name as "Last, First".    Args:      full\_name: The full name string (e.g., "Samreen Qhuddisia").  Returns:      The formatted name string (e.g., "Qhuddisia, Samreen").    """    # Assuming the full name is in the format "First Last"    name\_parts = full\_name.split()    if len(name\_parts) >= 2:      first\_name = name\_parts[0]      last\_name = name\_parts[-1] # Handle potential middle names by taking the last part as the last name      return f"{last\_name}, {first\_name}"    else:      # Handle cases where the name might be a single word or empty      return full\_name  format\_name("krithi Sanan")  Output: Sanan, krithi  **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  |  |  | | --- | --- | | **Zero-shot Prompt** A zero-shot prompt is straightforward. You simply ask the model to perform the task without any examples.  **Prompt:** Write a Python function that counts the number of vowels in a string. The vowels are 'a', 'e', 'i', 'o', 'u' | **Few-shot Prompt** A few-shot prompt provides a few examples to guide the modelThis is especially useful for tasks that are nuanced or require a specific output format.  **Prompt:** Write a Python function that counts the number of vowels in a string. The vowels are 'a', 'e', 'i', 'o', 'u'.**Example 1:** Input: "Programming" Output:3 **Example 2:** Input: "Data Science" Output: 5 | | **ef count\_vowels(s):**  **"""**  **Counts the number of vowels in a string.**  **"""**  **vowels = "aeiouAEIOU"**  **count = 0**  **for char in s:**  **if char in vowels:**  **count += 1**  **return count**  **# Example usage**  **my\_string = "Hello World"**  **print(f"The number of vowels in '{my\_string}' is: {count\_vowels(my\_string)}")**  **# Expected Output: The number of vowels in 'Hello World' is: 3** | **def count\_vowels(s):**  **"""**  **Counts the number of vowels in a string, case-insensitively.**  **"""**  **vowel\_count = 0**  **vowels = set("aeiou")**  **for char in s.lower():**  **if char in vowels:**  **vowel\_count += 1**  **return vowel\_count**  **# Example usage**  **my\_string = "Hello World"**  **print(f"The number of vowels in '{my\_string}' is: {count\_vowels(my\_string)}")**  **# Expected Output: The number of vowels in 'Hello World' is: 3** |   **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 .txt file and returns the number of lines.**Input:** file.txt with content: Hello, world! This is line 2. And this is line 3. **Output:** 3**Input:** empty\_file.txt with content: (empty) **Output:** 0 **Input:** single\_line.txt with content: Just one line.  def count\_lines(filepath):  """  Reads a file and returns the number of lines.  """  try:  with open(filepath, 'r') as file:  line\_count = 0  for line in file:  line\_count += 1  return line\_count  except FileNotFoundError:  print(f"Error: The file at '{filepath}' was not found.")  return -1  except Exception as e:  print(f"An error occurred: {e}")  return -1  # Example usage (assuming 'sample.txt' exists with some content)  # You can replace 'sample.txt' with the path to your file.  file\_path = "sample.txt"  num\_lines = count\_lines(file\_path)  if num\_lines != -1:  print(f"The file has {num\_lines} lines.")  **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 #5) | 1.0 | | Comparison (Task#4) | 0.5 | | **Total** | **2.5 Marks** | | | | | | | Week2 - Wednesday |  |