|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.   **Promt:**  **Write a function that checks whether a given year is a leap year**  **Code:**  def is\_leap\_year(year):    """    Checks if a given year is a leap year.    Args:      year: An integer representing the year.    Returns:      True if the year is a leap year, False otherwise.    """    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)  # Example usage:  print(f"2000 is a leap year: {is\_leap\_year(2000)}")  print(f"1900 is a leap year: {is\_leap\_year(1900)}")  print(f"2024 is a leap year: {is\_leap\_year(2024)}")  print(f"2023 is a leap year: {is\_leap\_year(2023)}")  **Expected Output#1**    **Task Description#2**   * One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.   **Promt:**  **Write a function that converts centimeters to inches**  **Code:**  def cm\_to\_inches(cm):    """    Converts centimeters to inches.    Args:      cm: A float representing the measurement in centimeters.    Returns:      A float representing the measurement in inches.    """    inches = cm \* 0.393701    return inches  # Example usage:  centimeters = 10  inches = cm\_to\_inches(centimeters)  print(f"{centimeters} centimeters is equal to {inches} inches")  **Expected Output#2**  **Task Description#3**   * Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.   **Promt:**  **Provide 2-3 examples to generate a function that formats full names as “Last, First”.**  **Code-1:**  import re  def format\_name\_example2(full\_name):    """    Formats a full name as "first,last" using regular expressions.    Args:      full\_name: A string representing the full name (e.g., "John Doe").    Returns:      A string representing the name in "first,last" format (e.g., "John,Doe").    """    match = re.match(r'^(\w+)\s+(.+)$', full\_name)    if match:      return f"{match.group(1)},{match.group(2)}"    else:      return full\_name # Handle cases that don't match the pattern  # Example usage:  print(format\_name\_example2("John Doe"))  print(format\_name\_example2("Jane Smith-Jones")) # Example with a hyphenated name  print(format\_name\_example2("Peter"))  **Code-2:**  def format\_name\_example1(full\_name):    """    Formats a full name as "first,last" by splitting and joining.    Args:      full\_name: A string representing the full name (e.g., "John Doe").    Returns:      A string representing the name in "first,last" format (e.g., "John,Doe").    """    name\_parts = full\_name.split()    if len(name\_parts) >= 2:      return f"{name\_parts[0]},{name\_parts[-1]}"    else:      return full\_name # Handle cases with only one name part  # Example usage:  print(format\_name\_example1("John Doe"))  print(format\_name\_example1("Jane Smith"))  print(format\_name\_example1("Peter")) # Example with one name  **Expected Output#3**  Examplke-1:    **Example-2:**    **Task Description#4**   * Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.   **Promt:**  **Write a function that counts the number of vowels in a string**  **Code:**  def count\_vowels(text):    """    Counts the number of vowels in a string.    Args:      text: The input string.    Returns:      The number of vowels in the string.    """    vowels = "aeiouAEIOU"    vowel\_count = 0    for char in text:      if char in vowels:        vowel\_count += 1    return vowel\_count  # Example usage:  text = "Hello World"  vowel\_count = count\_vowels(text)  print(f"The number of vowels in '{text}' is: {vowel\_count}")  **Expected Output#4**  **C:\Users\welcome\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot 2025-08-22 162328.png**  **Task Description#5**   * Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.   **Promt:**  **Generate a function that reads a .txt file and return the number of lines**  **Code:**    def count\_lines\_in\_file(file\_path):    """    Reads a text file and returns the number of lines.    Args:      file\_path: The path to the text file.    Returns:      The number of lines in the file, or -1 if the file is not found.    """    try:      with open(file\_path, 'r') as file:        line\_count = sum(1 for line in file)      return line\_count    except FileNotFoundError:      print(f"Error: File not found at {file\_path}")      return -1  # Example usage (replace "your\_file.txt" with your file path):  # file\_path = "your\_file.txt"  # num\_lines = count\_lines\_in\_file(file\_path)  # if num\_lines != -1:  #   print(f"The file '{file\_path}' has {num\_lines} lines.")  **Expected Output#5**  **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 |  |