|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 - Tuesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:4.2**(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 with only the instruction — Write a Python function to generate the Fibonacci sequence up to n terms   **Expected Output#1**   * A working function without using any sample inputs/outputs.   def fibonacci\_sequence(n):      """      Generates the Fibonacci sequence up to n terms.      The Fibonacci sequence is a series of numbers where each number is the sum      of the two preceding ones, usually starting with 0 and 1.        Args:          n (int): The number of terms in the sequence. Must be a non-negative integer.  Returns:          list: A list containing the Fibonacci sequence up to n terms.                Returns an empty list if n is less than or equal to 0.      """      # Handle edge case for n less than or equal to 0.      if n <= 0:          return []  # Handle the base case for n = 1.      elif n == 1:          return [0]  # Generate the sequence for n > 1.      else:          # Initialize the sequence with the first two terms.          sequence = [0, 1]            # Loop until the sequence has n terms.          while len(sequence) < n:              # Calculate the next term by adding the last two terms.              next\_term = sequence[-1] + sequence[-2]              # Add the new term to the sequence.              sequence.append(next\_term)            return sequence  print(fibonacci\_sequence(5))  **OUTPUT: [0, 1, 1, 2, 3]**  **Task Description#2**   * One-shot: Provide one example: Input: 100, Output: 37.78 to help AI generate a function that converts Fahrenheit to Celsius.   **Expected Output#2**   * A correct conversion function guided by the single example.   def fahrenheit\_to\_celsius(fahrenheit):      """      Converts a temperature from Fahrenheit to Celsius.      The formula for the conversion is (Fahrenheit - 32) \* 5/9.  Args:          fahrenheit (float or int): The temperature in Fahrenheit.  Returns:          float: The temperature converted to Celsius, rounded to two decimal places.      """      # Apply the conversion formula.      celsius = (fahrenheit - 32) \* 5/9        # Return the result, rounded to two decimal places.      return round(celsius, 2)  # Example usage based on the provided input.  # The user's example is converting 100°F.  input\_temp = 100  output\_temp = fahrenheit\_to\_celsius(input\_temp)  print(f"Input: {input\_temp}, Output: {output\_temp}")  OUTPUT: Input: 100, Output: 37.78  **Task Description#3**   * **Few-shot:** Give 2–3 examples to create a function that extracts the domain name from an email address.   **Expected Output#3**   * Accurate function that returns only the domain portion of an email (e.g., @gmail.com).   def get\_domain\_from\_email(email\_address):      """      Extracts the domain name from a given email address.      Args:          email\_address (str): The email address as a string.  Returns:          str: The domain portion of the email, including the '@' symbol.               Returns an empty string if '@' is not found.      """      # Find the index of the '@' symbol.      at\_symbol\_index = email\_address.find('@')  # Check if the '@' symbol was found.      if at\_symbol\_index != -1:          # Slice the string from the '@' symbol to the end to get the domain.          domain = email\_address[at\_symbol\_index:]          return domain      else:          # If no '@' symbol is found, return an empty string.          return ""  # Examples to demonstrate the function's usage.  email1 = "testuser@gmail.com"  email2 = "contact.me@my-business.org"  email3 = "newsletter@subdomain.web-host.net"  email4 = "no-at-sign"  print(f"'{email1}' -> '{get\_domain\_from\_email(email1)}'")  print(f"'{email2}' -> '{get\_domain\_from\_email(email2)}'")  print(f"'{email3}' -> '{get\_domain\_from\_email(email3)}'")  print(f"'{email4}' -> '{get\_domain\_from\_email(email4)}'")  OUTPUT: 'testuser@gmail.com' -> '@gmail.com' 'contact.me@my-business.org' -> '@my-business.org' 'newsletter@subdomain.web-host.net' -> '@subdomain.web-host.net' 'no-at-sign' -> ''  **Task Description#4**   * Compare zero-shot vs few-shot prompting for generating a function that checks whether a word is a palindrome, ignoring punctuation and case.   **Expected Output#4**   * Output comparison + student explanation on how examples helped the model.  |  |  | | --- | --- | | **Zero-shot prompting**  "Write a Python function to check if a word is a palindrome" often leads to the most basic interpretation of the request. It typically focuses on the core logic of reversing a string and comparing it to the original, without considering more complex requirements like handling punctuation or different cases. | **Few-shot prompting**  A few-shot prompt, on the other hand, provides concrete examples that highlight the need for a more complex solution. The examples 'RaceCar' and 'Madam, I'm Adam' tell the model that it must **normalize the input** by converting it to a consistent case and removing any non-alphanumeric characters *before* performing the check. | | def is\_palindrome\_zero\_shot(word):      """      Checks if a word is a palindrome using a basic string reversal.      This function is a simple implementation and does not handle      case sensitivity or punctuation.      Args:          word (str): The word to check.  Returns:          bool: True if the word is a palindrome, False otherwise.      """      # Simply compare the word to its reverse.  return word == word[::-1]  # Examples to show the function's behavior  print(f"Is 'racecar' a palindrome? {is\_palindrome\_zero\_shot('racecar')}")  print(f"Is 'RaceCar' a palindrome? {is\_palindrome\_zero\_shot('RaceCar')}")  print(f"Is 'A man, a plan, a canal: Panama' a palindrome? {is\_palindrome\_zero\_shot('A man, a plan, a canal: Panama')}")  OUTPUT:Is 'racecar' a palindrome? True Is 'RaceCar' a palindrome? False Is 'A man, a plan, a canal: Panama' a palindrome? False | import re  def is\_palindrome\_few\_shot(phrase):      """      Checks if a phrase is a palindrome by ignoring punctuation and case.      This function is more robust and handles more complex inputs.  Args:          phrase (str): The phrase to check.  Returns:          bool: True if the phrase is a palindrome, False otherwise.      """      # Step 1: Normalize the string.      # Convert to lowercase and remove all non-alphanumeric characters.      # The examples in the few-shot prompt guided the model to this crucial step.      normalized\_phrase = re.sub(r'[^a-zA-Z0-9]', '', phrase).lower()        # Step 2: Compare the normalized string to its reverse.      return normalized\_phrase == normalized\_phrase[::-1]  # Examples to show the function's improved behavior  print(f"Is 'racecar' a palindrome? {is\_palindrome\_few\_shot('racecar')}")  print(f"Is 'RaceCar' a palindrome? {is\_palindrome\_few\_shot('RaceCar')}")  print(f"Is 'A man, a plan, a canal: Panama' a palindrome? {is\_palindrome\_few\_shot('A man, a plan, a canal: Panama')}")  OUTPUT:Is 'racecar' a palindrome? True Is 'RaceCar' a palindrome? True Is 'A man, a plan, a canal: Panama' a palindrome? True |   **Task Description#5**   * Use few-shot prompting with 3 sample inputs to generate a function that determines the maximum of three numbers without using the built-in max() function.   **Expected Output#5**   * A function that handles all cases with correct logic based on example patterns.   def find\_maximum(a, b, c):      """      Determines the maximum of three numbers without using the built-in max() function.      The logic is based on a series of comparisons to find the largest value.      Args:          a (int or float): The first number.          b (int or float): The second number.          c (int or float): The third number.      Returns:          int or float: The largest of the three numbers.      """      # Assume 'a' is the largest initially.      # The few-shot examples demonstrate that we need to handle different number orders.      maximum = a      # Compare 'b' with the current maximum.      # If 'b' is larger, update the maximum.      if b > maximum:          maximum = b      # Compare 'c' with the current maximum.      # If 'c' is larger, update the maximum.      if c > maximum:          maximum = c      return maximum  # Demonstration of the function using the provided examples.  print(f"The maximum of 10, 5, 20 is: {find\_maximum(10, 5, 20)}")  print(f"The maximum of 3, 9, 3 is: {find\_maximum(3, 9, 3)}")  print(f"The maximum of 8, 8, 8 is: {find\_maximum(8, 8, 8)}")  print(f"The maximum of 8, 10, 8 is: {find\_maximum(8, 10, 8)}")  OUTPUT: The maximum of 10, 5, 20 is: 20 The maximum of 3, 9, 3 is: 9 The maximum of 8, 8, 8 is: 8 The maximum of 8, 10, 8 is: 10  **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 |  |