HT.NO:2303A52448

Batch-31

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | Dr. Rishabh Mittal | | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | Mr. S Naresh Kumar | | Ms. B. Swathi | | Dr. Sasanko Shekhar Gantayat | | Mr. Md Sallauddin | | Dr. Mathivanan | | Mr. Y Srikanth | | Ms. N Shilpa | | Dr. Rishabh Mittal (Coordinator) | | Dr. R. Prashant Kumar | | Mr. Ankushavali MD | | Mr. B Viswanath | | Ms. Sujitha Reddy | | Ms. A. Anitha | | Ms. M.Madhuri | | Ms. Katherashala Swetha | | Ms. Velpula sumalatha | | Mr. Bingi Raju | | | | | | |
| **CourseCode** | | | 23CS002PC304 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | III/II | **Regulation** | | R23 | | | |
| **Date and Day**  **of Assignment** | | | **Week2 – Monday** | **Time(s)** | | 23CSBTB01 To 23CSBTB52 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All 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, Chain of Thoughts, Prompt Chaining, Directional stimulus prompting**  **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.   **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. | | | | | |  |  |
|  |  | Q1. Zero-Shot Prompting (Basic Lab Task)  Task: Write a Python function that classifies a given text as Spam or Not Spam using zero-shot prompting.  Steps:   1. Construct a prompt without any examples.   Classify the following given text as spam or not spam and display the result as spam or not spam   1. Clearly specify the output labels.   # Classify the following given text as spam or not spam and display the result as spam or not spam  def classify\_text(text):  spam\_keywords = ['win', 'free', 'prize', 'click', 'buy now', 'limited time offer', 'congratulations']  text\_lower = text.lower()    for keyword in spam\_keywords:  if keyword in text\_lower:  return "spam"    return "not spam"  # Example usage  input\_text = "Congratulations! You have won a free prize. Click here to claim it."  result = classify\_text(input\_text)  print(result) # Output: spam  # Example usage  input\_text2 = "Hello, I hope you are having a great day!"  result2 = classify\_text(input\_text2)  print(result2) # Output: not spam  # Example usage  input\_text3 = "Limited time offer! Buy now and save big."  result3 = classify\_text(input\_text3)  print(result3) # Output: spam  # Example usage  input\_text4 = "Congratulations! You have won a free lottery ticket."  result4 = classify\_text(input\_text4)  print(result4) # Output: spam   1. Display only the predicted label.     Explanation: If a message contains any keyword from the list spam\_keywords = ['win', 'free', 'prize', 'click', 'buy now', 'limited time offer', 'congratulations'], it is detected as spam; otherwise, it is classified as not spam.  Input: "Congratulations! You have won a free lottery ticket."  Expected Output: Spam  Q2. One-Shot Prompting (Emotion detection)  Task: Write a Python program that detects the emotion of a sentence using one-shot prompting.  Emotions: [‘happy', 'sad', 'angry', 'excited', 'nervous', ’neutral’]  Steps:   1. Provide one labeled example inside the prompt.   from the given list of Emotions: [‘happy', 'sad', 'angry', 'excited', 'nervous', ’neutral’]  example : I am so thrilled about the upcoming trip!  output : excited  now classify the emotions based on sentences and display the result as one of the emotions from the list.   1. Take a sentence as input.   # Example usage  input\_text = "I am so thrilled about the upcoming trip!"  result = classify\_emotion(input\_text)  print(result) # Output: excited  # Example usage  input\_text2 = "I feel very anxious before my presentation."  result2 = classify\_emotion(input\_text2)  print(result2) # Output: nervous   1. Print the predicted emotion     Explanation: From the given list of emotions happy, sad, angry, excited, nervous, and neutral the sentence “I am so thrilled about the upcoming trip!” expresses strong enthusiasm and joy. it display *excited*.  Q3. Few-Shot Prompting (Student Grading Based on Marks)  Task: Write a Python program that predicts a student’s grade based on marks using few-shot prompting.  Grades:  ['A', 'B', 'C', 'D', 'F']  Grading Criteria (to be inferred from examples):   * 90–100 → A * 80–89 → B * 70–79 → C * 60–69 → D * Below 60 → F   '''create a grading system  example  A - 95  B - 85  C - 75  D - 60  F - below 60  '''  def grade\_system(score):  if score >= 95:  return 'A'  elif score >= 85:  return 'B'  elif score >= 75:  return 'C'  elif score >= 60:  return 'D'  else:  return 'F'  # Example usage  input\_score = 88  result = grade\_system(input\_score)  print(result) # Output: B  # Example usage  input\_score2 = 72  result2 = grade\_system(input\_score2)  print(result2) # Output: C  # Example usage  input\_score3 = 59  result3 = grade\_system(input\_score3)  print(result3) # Output: F  Output    **Q4. Multi-Shot Prompting (Indian Zodiac Sign Prediction using Month Name)**  Task: Write a Python program that predicts a person’s Indian Zodiac sign (Rashi) based on the month of birth (month name) using multi-shot prompting.  Indian Zodiac Order (Simplified Month-Based Model): The Indian Zodiac cycle starts in March with Mesha and follows this order:  March → Mesha April → Vrishabha May → Mithuna June → Karka July → Simha August → Kanya September → Tula October → Vrischika November → Dhanu December → Makara January → Kumbha February → Meena  '''create a predicts a person’s Indian Zodiac sign based on their birth year  March → Mesha  April → Vrishabha  May → Mithuna  June → Karka  July → Simha  August → Kanya  September → Tula  October → Vrischika  November → Dhanu  December → Makara  January → Kumbha  February → Meena  example  input: March  output: Mesha  input: May  output: Mithuna  '''  def indian\_zodiac\_sign(month):  zodiac\_signs = {  'march': 'Mesha',  'april': 'Vrishabha',  'may': 'Mithuna',  'june': 'Karka',  'july': 'Simha',  'august': 'Kanya',  'september': 'Tula',  'october': 'Vrischika',  'november': 'Dhanu',  'december': 'Makara',  'january': 'Kumbha',  'february': 'Meena'  }    month\_lower = month.lower()  return zodiac\_signs.get(month\_lower, "Invalid month")  # Example usage  input\_month = "March"  result = indian\_zodiac\_sign(input\_month)  print(result) # Output: Mesha  # Example usage  input\_month2 = "May"  result2 = indian\_zodiac\_sign(input\_month2)  print(result2) # Output: Mithuna  Output    **Q5. Result Analysis Based on Marks**  Task: Write a Python program that determines whether a student Passes or Fails based on marks using Chain-of-Thought (CoT) prompting.  Result Categories:  ['Pass', 'Fail']  '''read marks from students  check if marks is greater than or equal to 40  if yes print pass  otherwise print fail  well commented code'''  marks = float(input("Enter the marks obtained by the student: ")) # Read marks from user and convert to float  # Check if the marks are greater than or equal to 40  if marks >= 40:  print("Pass") # Print "Pass" if marks are 40 or more  else:  print("Fail") # Print "Fail" if marks are less than 40    Output    **Q6 Voting Eligibility Check (Chain-of-Thought Prompting)**  Task: Write a Python program that determines whether a person is eligible to vote using Chain-of-Thought (CoT) prompting.  '''read age from user to check voting eligibility  check if age is greater than or equal to 18  if yes print eligible to vote  otherwise print not eligible to vote  well commented code'''  age = int(input("Enter your age: ")) # Read age from user and convert to integer  # Check if the age is greater than or equal to 18  if age >= 18:  print("Eligible to vote") # Print if the user is eligible to vote  else:  print("Not eligible to vote") # Print if the user is not eligible to vote  Output    **Q7 Prompt Chaining (String Processing – Palindrome Names)**  Task: Write a Python program that uses the prompt chaining technique to identify palindrome names from a list of student names.  '''create a list of students with thier names give own names  from the list find the student name is palindrome or not  if yes print palindrome student name  otherwise print not palindrome student name  '''  def is\_palindrome(name):  # Check if the name is the same forwards and backwards  return name == name[::-1]  students = ["Anna", "Bob", "Cathy", "David", "Eve"] # List of student names  for student in students:  if is\_palindrome(student):  print(f"{student} is a palindrome student name.") # Print if the name is a palindrome  else:  print(f"{student} is not a palindrome student name.") # Print if the name is not a palindrome  Output    **Q8 Prompt Chaining (String Processing – Word Length Analysis)**  **Task:** Write a Python program that uses **prompt chaining** to analyze a list of words. In the first prompt, generate a list of words. In the second prompt, traverse the list and calculate the length of each word. In the third prompt, use the output of the previous step to determine whether each word is **Short** (length less than 5) or **Long** (length greater than or equal to 5), and display the result for each word  # Generate a list of words.  words = ["level", "world", "radar", "python", "civic", "java", "deified", "code", "rotor", "script"]  # Calculate length of each word  for word in words:  length = len(word) # Calculate the length of the word  print(f"The length of the word '{word}' is {length}.") # Print the word and its length  # Classify words as Short or Long  if length < 5:  print(f"'{word}' is a Short word.") # Print if the word is short  else:  print(f"'{word}' is a Long word.") # Print if the word is long  print() # Print a newline for better readability  Output | | | | | |  |  |