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.	Week2 - Monday

- 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

Expected Output

- A valid Python function that performs all required validations without using any input-output examples in the prompt.

Prompt: Write a python function which read a number that validates an Indian mobile number and check whether that contains exactly 10 digits and Starts with 6,7,8,or 9.

task1.py

```python
def is_valid_indian_mobile(number):
    """
    Validates if the input is a valid Indian mobile number:
    - Exactly 10 digits
    - Starts with 6, 7, 8, or 9
    """
    num_str = str(number).strip()
    return len(num_str) == 10 and num_str.isdigit() and num_str[0] in {'6', '7', '8', '9'}

mobile = input("Enter mobile number: ")
if is_valid_indian_mobile(mobile):
    print(f"{mobile} is a valid Indian mobile number")
else:
    print(f"{mobile} is an invalid Indian mobile number")
```

Output:

```
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyt
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task1.py
Enter mobile number: 8309808491
8309808491 is a valid Indian mobile number
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyt
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task1.py
Enter mobile number: 6789087654433
6789087654433 is an invalid Indian mobile number
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyt
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task1.py
Enter mobile number: 1234567891
1234567891 is an invalid Indian mobile number
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

task1_gc:

task1gc.py > ...

```python
def is_valid_indian_mobile(number):
    """
    Validates if the input string is a valid Indian mobile number.
    - Must be exactly 10 digits.
    - Must start with 6, 7, 8, or 9.
    """
    if len(number) == 10 and number.isdigit() and number[0] in '6789':
        return True
    return False

# Example usage:
mobile = input("Enter mobile number: ")
if is_valid_indian_mobile(mobile):
    print("Valid Indian mobile number.")
else:
    print("Invalid Indian mobile number.")
```

Output:

```
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Pr
ograms/Python/Python313/python.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task1gc.p
y
Enter mobile number: 9700585209
Valid Indian mobile number.
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

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

Expected Output

- A Python function with correct factorial logic and edge case handling, generated from a single example.

Prompt: Write a python program using functions that calculates the factorial of a number for positive integer only if negative return an appropriate message.

Example: 0! = 0

```python
def factorial(n):
    """
    Calculates the factorial of a non-negative integer n.
    Returns an error message if n is negative or not an integer.
    By convention, 0! = 0 as per the example.
    """
    if not isinstance(n, int):
        return "Input must be an integer."
    if n < 0:
        return "Factorial is not defined for negative numbers."
    if n == 0:
        return 0
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

try:
    num = int(input("Enter a positive integer: "))
    res = factorial(num)
    if isinstance(res, str):
        print(res)
    else:
        print(f"{num}! = {res}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

Output:

```
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyt
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task2.py
Enter a positive integer: 5
5! = 120
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyt
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task2.py
Enter a positive integer: -5
Factorial is not defined for negative numbers.
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyt
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task2.py
Enter a positive integer: 0
0! = 0
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

Task2_gc:

```python
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0:
        return 0
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result

# Example usage
num = int(input("Enter a positive integer: "))
fact = factorial(num)
print(f"{num}! = {fact}")
```

Output:

```
Enter a positive integer: 4
4! = 24
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Pr
ograms/Python/Python313/python.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task2gc.p
y
Enter a positive integer: -4
-4! = Factorial is not defined for negative numbers.
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Pr
ograms/Python/Python313/python.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task2gc.p
y
Enter a positive integer: 0
0! = 0
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

	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:
	  o Full Name
	  o Branch
	  o SGPA
	Expected Output
	• A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples.
	Prompt: Write a python program using nested dictionary Extraction representing student information where the function should extract and return FULL NAME, BRANCH, SGPA
	Example: FULL NAME: xxxx Sirichandana
	BRANCH: CSE
	SGPA: 9.2345677
	Example: FULL NAME: xxxx Hasini
	BRANCH: ECE
	SGPA: 9.9244558
	Example: FULL NAME: xxxx Nivedha
	BRANCH: CSE
	SGPA: 9.5344678	

```python
def extract_student_info(student_dict):
    Extracts and returns FULL NAME, BRANCH, and SGPA from a nested student dictionary.
    """
    full_name = student_dict.get('FULL NAME', 'N/A')
    branch = student_dict.get('BRANCH', 'N/A')
    sgpa = student_dict.get('SGPA', 'N/A')
    return full_name, branch, sgpa
# Example nested dictionary of students
students = {
    1: {
        'FULL NAME': 'xxxx Sirichandana',
        'BRANCH': 'CSE',
        'SGPA': 9.2345677
    },
    2: {
        'FULL NAME': 'xxxx Hasini',
        'BRANCH': 'ECE',
        'SGPA': 9.9244558
    },
    3: {
        'FULL NAME': 'xxxx Nivedha',
        'BRANCH': 'CSE',
        'SGPA': 9.5344678
    }
}
# Extract and print information for each student
for student_id, info in students.items():
    full_name, branch, sgpa = extract_student_info(info)
    print(f"FULL NAME: {full_name}")
    print(f"BRANCH: {branch}")
    print(f"SGPA: {sgpa}")
    print()  # Blank line between students
```

Output:

```
hon.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task3.py
FULL NAME: xxxx Sirichandana
BRANCH: CSE
SGPA: 9.2345677

FULL NAME: xxxx Hasini
BRANCH: ECE
SGPA: 9.9244558

FULL NAME: xxxx Nivedha
BRANCH: CSE
SGPA: 9.5344678

PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

Task3_gc:

```python
# Nested dictionary representing student information
students = {
    1: {
        'first_name': 'xxxx',
        'last_name': 'Sirichandana',
        'branch': 'CSE',
        'sgpa': 9.2345677
    },
    2: {
        'first_name': 'xxxx',
        'last_name': 'Hasini',
        'branch': 'ECE',
        'sgpa': 9.9244558
    },
    3: {
        'first_name': 'xxxx',
        'last_name': 'Nivedha',
        'branch': 'CSE',
        'sgpa': 9.5344678
    }
}

def extract_student_info(student):
    full_name = f"{student['first_name']} {student['last_name']}"
    branch = student['branch']
    sgpa = student['sgpa']
    return full_name, branch, sgpa

# Extract and print information for each student
for sid, info in students.items():
    full_name, branch, sgpa = extract_student_info(info)
    print(f"FULL NAME: {full_name}")
    print(f"BRANCH: {branch}")
    print(f"SGPA: {sgpa}\n")
```

Output:

```
y
FULL NAME: xxxx Sirichandana
BRANCH: CSE
SGPA: 9.2345677

FULL NAME: xxxx Hasini
BRANCH: ECE
SGPA: 9.9244558

FULL NAME: xxxx Nivedha
BRANCH: CSE
SGPA: 9.5344678

PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

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

Expected Output

- Working Python functions for each prompting style, with a brief reflection comparing their accuracy, clarity, and efficiency.

Prompt1: *Write a Python function that reads a CSV file and returns the total number of rows, the number of empty rows, and the total number of words in the file.*

task4_1.py > analyze_csv

```python
import csv

def analyze_csv(file_path):
    total_rows = 0
    empty_rows = 0
    total_words = 0

    with open(file_path, newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            total_rows += 1
            # Check if all cells in the row are empty or whitespace
            if all(cell.strip() == '' for cell in row):
                empty_rows += 1
            # Count words in all cells
            for cell in row:
                total_words += len(cell.strip().split())

    return total_rows, empty_rows, total_words
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyth
on.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task4_1.py
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

Prompt2: Here's an example of a function that reads a CSV and returns the number of rows. Now write one that also counts empty rows and total words.

Example:

```
def count_rows(file_path):
    with open(file_path, newline='', encoding='utf-8') as f:
     return sum(1 for _ in csv.reader(f))
```

task4_2.py > count_rows_and_words

```python
import csv

def count_rows_and_words(file_path):
    total_rows = 0
    empty_rows = 0
    total_words = 0
    with open(file_path, newline='', encoding='utf-8') as f:
        reader = csv.reader(f)
        for row in reader:
            total_rows += 1
            if all(cell.strip() == '' for cell in row):
                empty_rows += 1
            total_words += sum(len(cell.split()) for cell in row)
    return total_rows, empty_rows, total_words
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyth
on.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task4_1.py
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyth
on.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task4_2.py
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

Prompt3: *Here are two examples of functions that read CSV files and perform analysis. Now write one that returns total rows, empty rows, and word count.*

Examples Given:

1. Count rows and columns
2. Count non-empty cells

task4_3.py > analyze_csv

```python
import csv

def analyze_csv(file_path):
    total_rows = 0
    empty_rows = 0
    word_count = 0

    with open(file_path, newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            total_rows += 1
            # Check if all cells in the row are empty
            if all(cell.strip() == '' for cell in row):
                empty_rows += 1
            # Count words in all cells of the row
            for cell in row:
                word_count += len(cell.split())

    return total_rows, empty_rows, word_count
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/pyth
on.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task4_3.py
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

Zero-shot works well for simple tasks but may lack polish or consistency.

- **One-shot** benefits from context and tends to produce cleaner logic.
- **Few-shot** shines when clarity and structure matter—especially for maintainable code.

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

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

Prompt: Write a Python function that takes a paragraph of text and finds the most common word. The function should:

- **Turn all the text into lowercase**
- **Remove any punctuation (like commas, periods, question marks, etc.)**
- **Split the text into words**
- **Count how often each word appears**
- **Return the word that appears the most**

Example:

```
def to_lowercase(text):
  return text.lower()
```

Example:

```
import string
def remove_punctuation(text):
  return text.translate(str.maketrans('', '',
string.punctuation))
Example: from collections import Counter
def count_words(text):
   words = text.split()
  return Counter(words)
```

```python
import string
from collections import Counter

def most_common_word(paragraph):
    # Convert to lowercase
    paragraph = paragraph.lower()
    # Remove punctuation
    paragraph = paragraph.translate(str.maketrans('', '', string.punctuation))
    # Split into words
    words = paragraph.split()
    # Count word frequencies
    word_counts = Counter(words)
    # Find the most common word
    if word_counts:
        return word_counts.most_common(1)[0][0]
    else:
        return None
paragraph = "Hello world! Hello again. This world is full of wonders, and hello to all."
print(most_common_word(paragraph))
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4> & C:/Users/Sirichandana/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Sirichandana/OneDrive/Desktop/AIAC/lab4/task5.py
hello
PS C:\Users\Sirichandana\OneDrive\Desktop\AIAC\lab4>
```

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**