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		
Course Code	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/I	**Regulation**	R23
Date and Day of Assignment	Week 2 - Wednesday	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicableto Batches**	All batches
Assignment Number: 3.3(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 • 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	Week 2 - Wednesday

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 different prompting strategies

Task 1: Zero-Shot Prompting – Leap Year Check
Scenario
Zero-shot prompting involves giving instructions without providing examples.
Task Description
Use zero-shot prompting to instruct an AI tool to generate a Python function that:

- Accepts a year as input
- Checks whether the given year is a leap year
- Returns an appropriate result

Note: No input-output examples should be provided in the prompt.
Expected Output

- AI-generated leap year checking function
- Correct logical conditions
- Sample input and output
- Screenshot of AI-generated response (if required)

Prompt:
Generate a Python function that accepts a year as input, checks whether the year is a leap year using correct logical conditions, and returns an appropriate result.

Code:

```
# Generate a Python function that accepts a year as input,
# checks whether the year is a leap year using correct logical
# conditions, and returns an appropriate result. user input: year
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage:
year = int(input("Enter a year: "))
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Output :

```
PS C:\Users\abhir\OneDrive\Desktop\ai> python -u "c:\
.py"
Enter a year: 2000
2000 is a leap year.
```

Explanation:
A year is a leap year if:

- **It is divisible by 400, OR**
- **It is divisible by 4 but not divisible by 100**

Task 2: One-Shot Prompting – Centimeters to Inches Conversion

Scenario

One-shot prompting guides AI using a single example.

Task Description

Use one-shot prompting by providing one input-output example to generate a Python function that:

- Converts centimeters to inches
- Uses the correct mathematical formula

Example provided in prompt:

Input: 10 cm → Output: 3.94 inches

Expected Output

- Python function with correct conversion logic
- Accurate calculation
- Sample test cases and outputs

Prompt:

Generate a Python function that converts centimeters to inches.

Code:

```
# Generate a Python function that converts centimeters to inches.
def cm_to_inches(cm):
    return cm / 2.54
cm = float(input("Enter length in centimeters: "))
inches = cm_to_inches(cm)
print(f"{cm} cm is equal to {inches} inches.")
```

Output:

```
PS C:\Users\abhir\OneDrive\Desktop\ai> python -u "c:\Users\a
.py"
Enter length in centimeters: 10
10.0 cm is equal to 3.937007874015748 inches.
```

Explanation:

- **The function converts centimeters to inches using the standard formula: inches = centimeters ÷ 2.54.**
- **User input is taken as a floating-point number to support decimal values.**
- **The converted value is printed directly, ensuring accurate and real-time conversion.**

Task 3: Few-Shot Prompting – Name Formatting

Scenario

Few-shot prompting improves accuracy by providing multiple examples.

Task Description

Use few-shot prompting with 2–3 examples to generate a Python function that:

- Accepts a full name as input
- Formats it as “Last, First”

Example formats:

- "John Smith" → "Smith, John"
- "Anita Rao" → "Rao, Anita"

Expected Output

- Well-structured Python function
- Output strictly following example patterns
- Correct handling of names
- Sample inputs and outputs

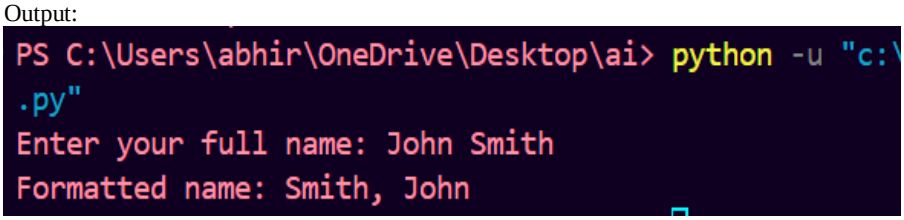
Prompt:

Generate a Python function that accepts a full name as input and formats it as “Last, First”.

Code:

```python
# Generate a Python function that accepts a full name as input and formats it as "Last, First".
def format_name(full_name):
    parts = full_name.split()
    if len(parts) >= 2:
        first_name = parts[0]
        last_name = parts[-1]
        return f"{last_name}, {first_name}"
    else:
        return "Invalid name format."
full_name = input("Enter your full name: ")
formatted_name = format_name(full_name)
print(f"Formatted name: {formatted_name}")
```

Output:



Explanation:

- The input name is split into individual words using spaces, allowing separation of first and last names.
- If at least two words are present, the first word is treated as the first name and the last word as the last name.
- The name is formatted as **"Last, First"**; otherwise, an error message is returned for invalid input.

Task 4: Comparative Analysis – Zero-Shot vs Few-Shot

Scenario

Different prompt strategies may produce different code quality.

Task Description

- Use zero-shot prompting to generate a function that counts vowels in a string
- Use few-shot prompting for the same problem
- Compare both outputs based on:
  - Accuracy
  - Readability
  - Logical clarity

Expected Output

- Two vowel-counting functions
- Comparison table or short reflection paragraph
- Conclusion on prompt effectiveness

Prompt:

Generate a Python function that counts the number of vowels in a given string and returns the count.

Code:

```python
# Generate a Python function that counts the number of vowels in a
# given string and returns the count.
def count_vowels(input_string):
    vowels = "aeiouAEIOU"
    count = 0
    for char in input_string:
        if char in vowels:
            count += 1
    return count
# Example usage:
input_string = input("Enter a string: ")
vowel_count = count_vowels(input_string)
print(f"Number of vowels in the string: {vowel_count}")
```

Output:

```
PS C:\Users\abhir\OneDrive\Desktop\ai> python -u "c:\Users\abh
.py"
Enter a string: hello
Number of vowels in the string: 2
```

Explanation:

- **A string containing all uppercase and lowercase vowels is defined to ensure case-insensitive checking.**
- **The function iterates through each character of the input string and checks whether it is a vowel.**
- **Each time a vowel is found, the counter is incremented, and the final count is returned.**

Task 5: Few-Shot Prompting – File Handling

Scenario

File processing requires clear logical understanding.

Task Description

Use few-shot prompting to generate a Python function that:

- Reads a .txt file
- Counts the number of lines in the file
- Returns the line count

Expected Output

- Working Python file-processing function
- Correct line count
- Sample .txt input and output
- AI-assisted logic explanation

Prompt:

Generate a Python function that reads a text file and counts the number of lines in it.

Code:

```python
# Generate a Python function that reads a text file and counts the number of lines in it.
def count_lines_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)
    except FileNotFoundError:
        return "File not found."
# Example usage:
file_path = input("Enter the file path: ")
line_count = count_lines_in_file(file_path)
print(f"Number of lines in the file: {line_count}")
```

Output:

```
PS C:\Users\abhir\OneDrive\Desktop\ai> python -u "c:\Users\abhir\OneDrive\Deskto
.py"
Enter the file path: C:\Users\abhir\OneDrive\Desktop\ai\some.txt
Number of lines in the file: 1
```

Explanation:

- The function attempts to open the given file path in read mode using a try block to handle possible errors.
- If the file is successfully opened, all lines are read and the total number of lines is calculated using len().
- If the file does not exist, the FileNotFoundError is caught and an appropriate message is returned instead of crashing the program.

	Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.	