NAME :A.SHASHIDHAR **H.NO:2303A51798** **BATCH:26**

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	Week2 - Wednesday

Lab Outcomes (LOs)
After completing this lab, students will be able to:

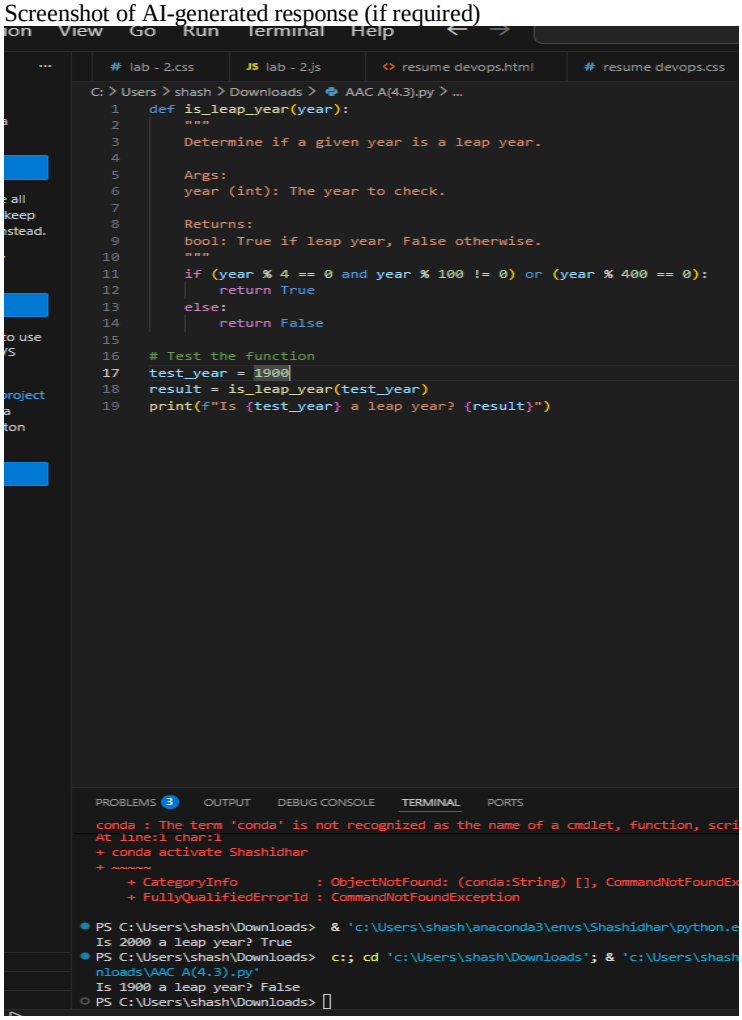
- Use zero-shot prompting to instruct AI with minimal context
- Use one-shot prompting with a single example to guide AI code generation
- Apply few-shot prompting using multiple examples to improve AI responses
- Compare AI outputs across different prompting strategies

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)



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

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:

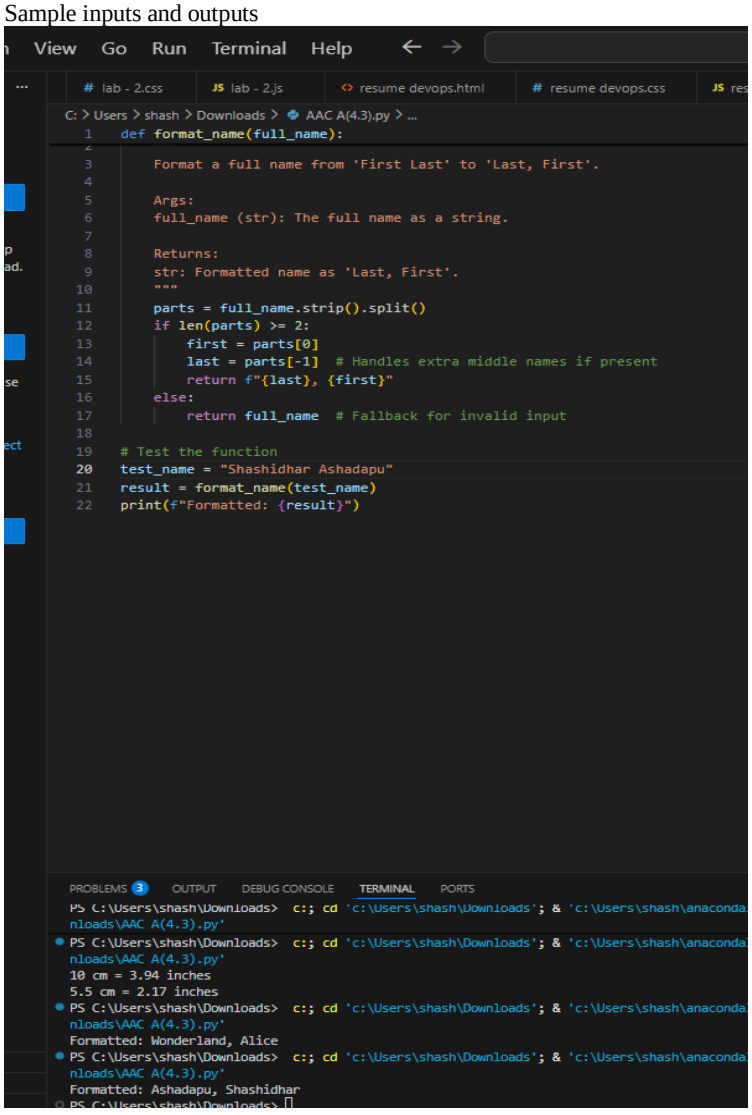
- Accepts a full name as input
- Formats it as “Last, First”

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



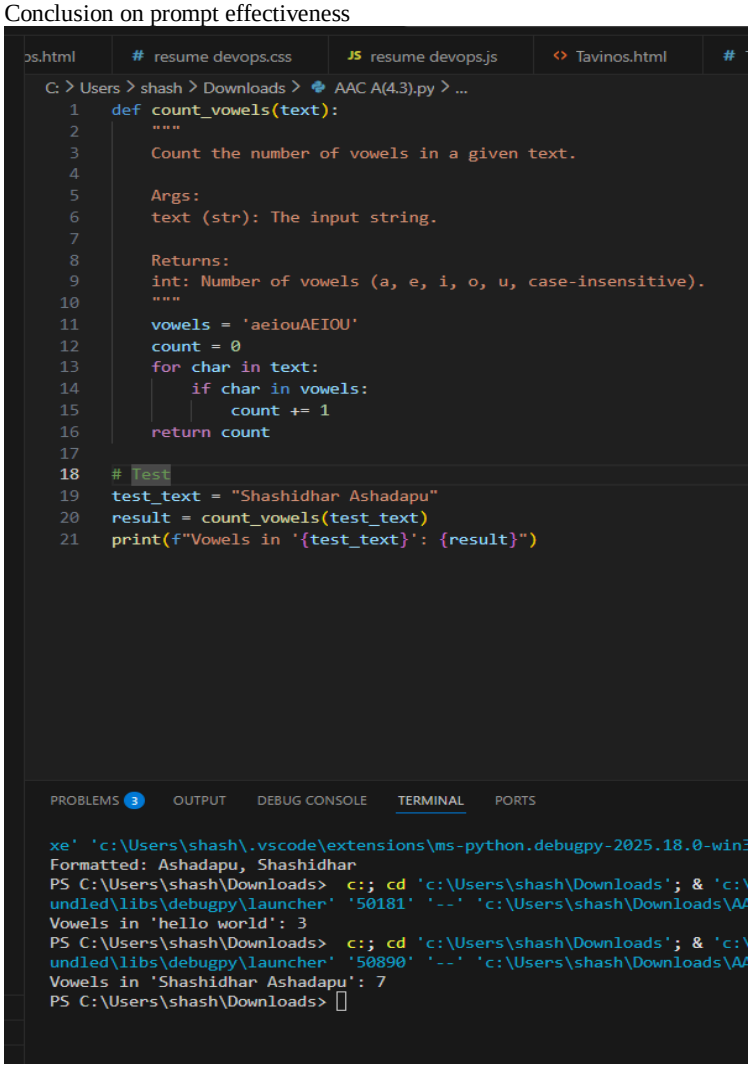
Task 4: Comparative Analysis – Zero-Shot vs Few-Shot
Scenario
Different prompt strategies may produce different code quality.
Task Description

- 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



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

NAME :A.SHASHIDHAR H.NO:2303A51798 BATCH:26

```
C: > Users > shash > Downloads > AAC A(4.3).py > ...
def count_lines_in_file(file_path):
    """
    Count the number of lines in a text file.

    Args:
    file_path (str): Path to the .txt file.

    Returns:
    int: Number of lines, or 0 if file not found.

    Raises:
    FileNotFoundError: If the file doesn't exist.
    """
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            return len(lines)
    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
        return 0

# Create a sample file for testing (run this once)
with open('sample.txt', 'w') as f:
    f.write("Line1\nLine2\nLine3\nLine4\nLine5")

# Test the function
result = count_lines_in_file('sample.txt')
print(f"Line count in 'sample.txt': {result}")
```

```
PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

undled\libs\debugpy\launcher' '50181' '--' 'c:\Users\shash\Downloads\AAC
Vowels in 'hello world': 3
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\U
undled\libs\debugpy\launcher' '50890' '--' 'c:\Users\shash\Downloads\AAC
Vowels in 'Shashidhar Ashadapu': 7
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\U
undled\libs\debugpy\launcher' '61017' '--' 'c:\Users\shash\Downloads\AAC
Line count in 'sample.txt': 3
PS C:\Users\shash\Downloads>  c:; cd 'c:\Users\shash\Downloads'; & 'c:\U
undled\libs\debugpy\launcher' '64519' '--' 'c:\Users\shash\Downloads\AAC
Line count in 'sample.txt': 5
PS C:\Users\shash\Downloads>
```

- **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.**