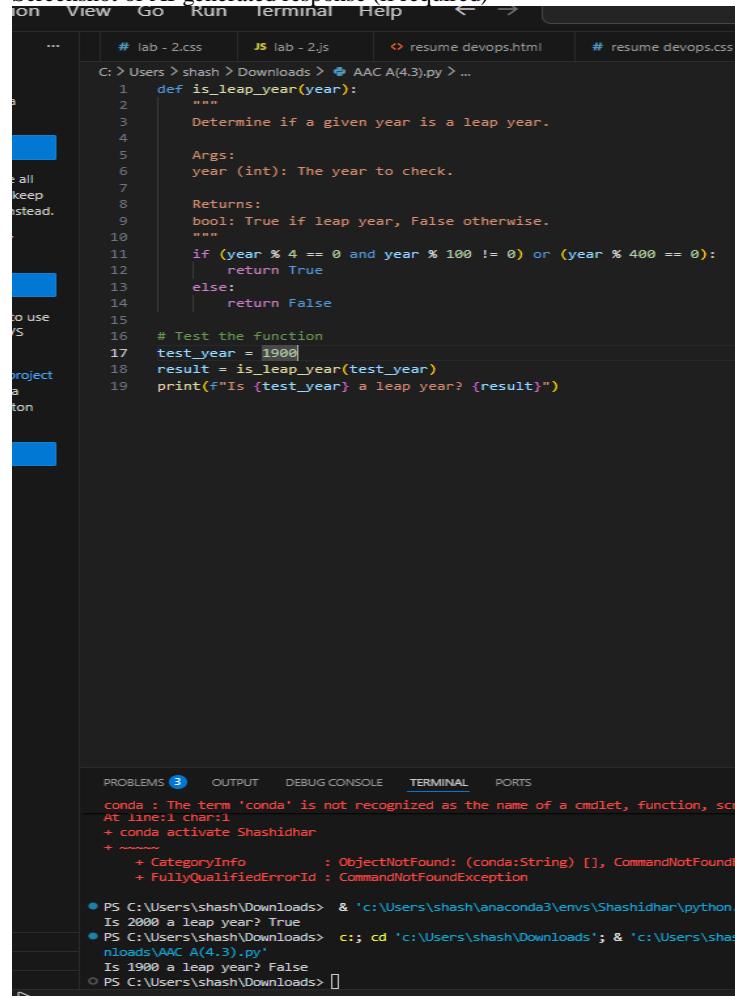
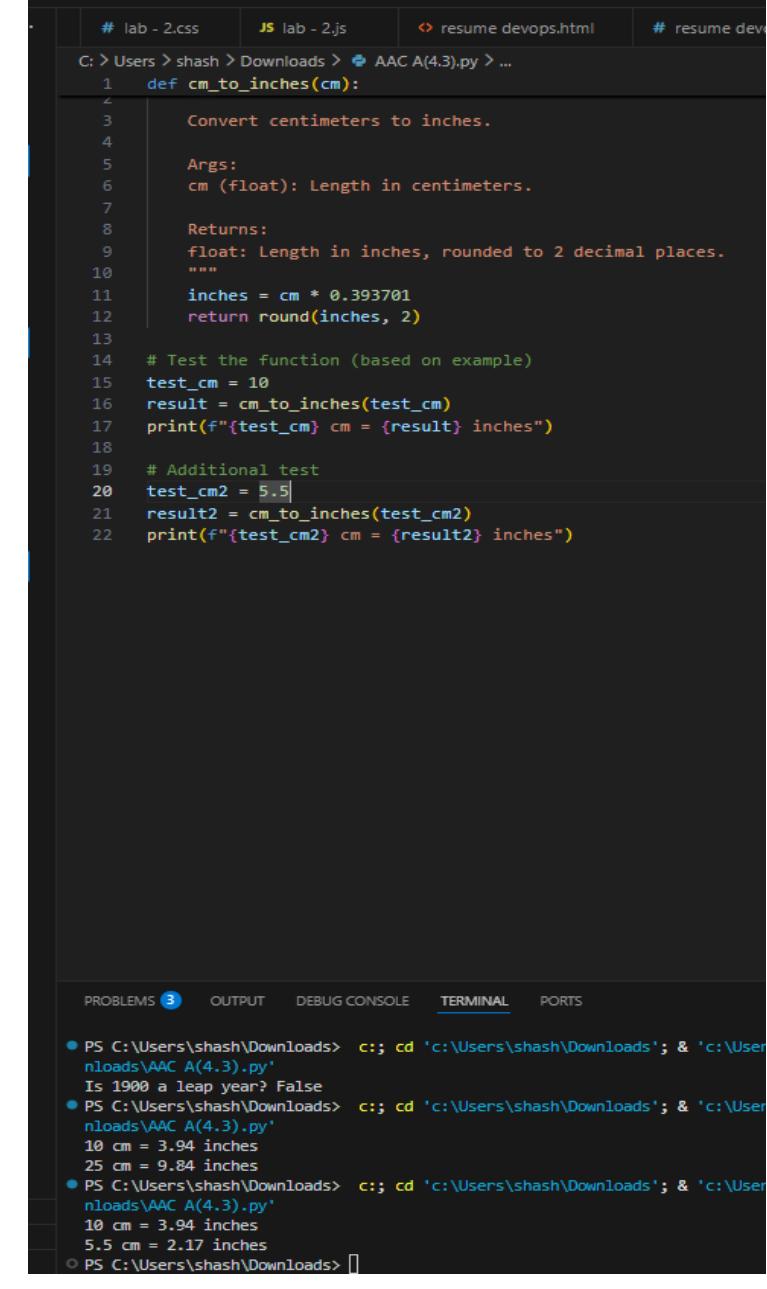
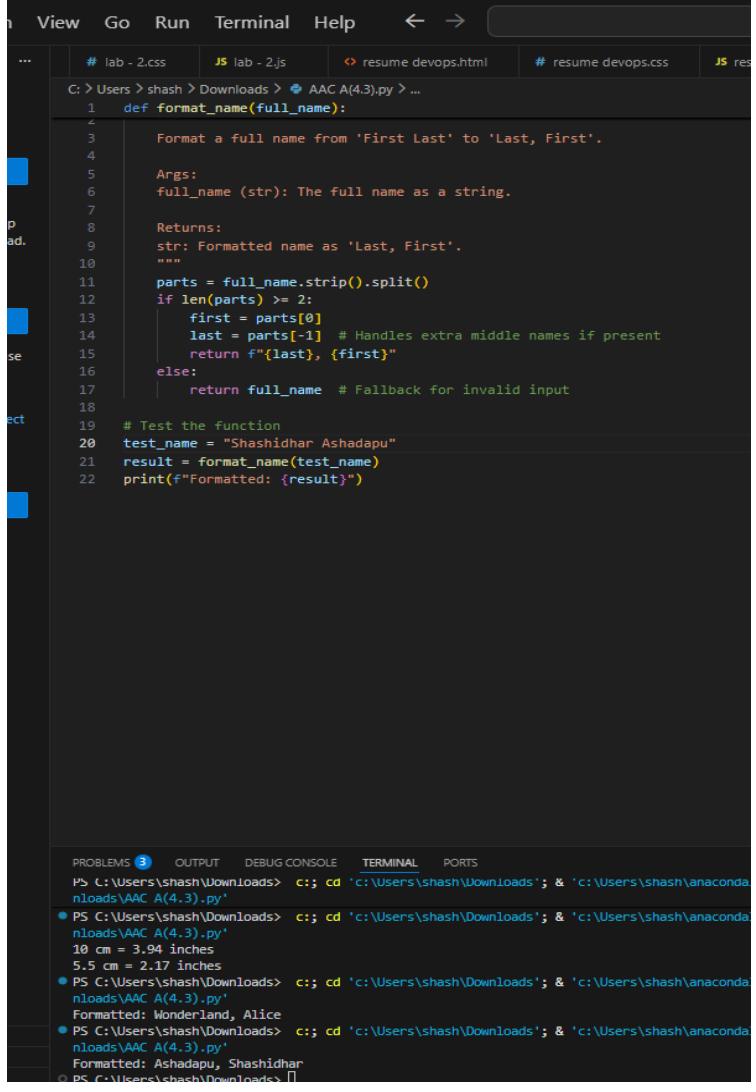


SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
<b>Instructor(s) Name</b>		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. 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	
<b>Course Code</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/I	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week 2 - Wednesday	<b>Time(s)</b>	23CSBTB01 To 23CSBTB52
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	All batches
<b>Assignment Number:</b> 3.3(Present assignment number)/24(Total number of assignments)			

<b>Q.No.</b>	<b>Question</b>	<b>Expected Time to complete</b>
1	<p><b>Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques</b></p> <p><b>Lab Objectives</b></p> <ul style="list-style-type: none"> <li>To explore and apply different levels of prompt examples in AI-assisted code generation</li> <li>To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality</li> <li>To evaluate the impact of context richness and example quantity on AI performance</li> <li>To build awareness of prompt strategy effectiveness for different problem types</li> </ul>	Week2 - Wednesday

	<p><b>Lab Outcomes (LOs)</b></p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> <li>• Use zero-shot prompting to instruct AI with minimal context</li> <li>• Use one-shot prompting with a single example to guide AI code generation</li> <li>• Apply few-shot prompting using multiple examples to improve AI responses</li> <li>• Compare AI outputs across different prompting strategies</li> </ul> <hr/> <p><b>Task 1: Zero-Shot Prompting – Leap Year Check</b></p> <p><b>Scenario</b></p> <p>Zero-shot prompting involves giving instructions without providing examples.</p> <p><b>Task Description</b></p> <p>Use zero-shot prompting to instruct an AI tool to generate a Python function that:</p> <ul style="list-style-type: none"> <li>• Accepts a year as input</li> <li>• Checks whether the given year is a leap year</li> <li>• Returns an appropriate result</li> </ul> <p><b>Note:</b> No input-output examples should be provided in the prompt.</p> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>• AI-generated leap year checking function</li> <li>• Correct logical conditions</li> <li>• Sample input and output</li> <li>• Screenshot of AI-generated response (if required)</li> </ul>  <pre>     ...     # lab - 2.css          JS lab - 2.js      resume devops.html  # resume devops.css C: &gt; Users &gt; shash &gt; Downloads &gt; AAC A(4.3).py &gt; ... 1 def is_leap_year(year): 2 """ 3     Determine if a given year is a leap year. 4 5     Args: 6         year (int): The year to check. 7 8     Returns: 9         bool: True if leap year, False otherwise. 10    """ 11    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0): 12        return True 13    else: 14        return False 15 16 # Test the function 17 test_year = 1900 18 result = is_leap_year(test_year) 19 print(f"Is {test_year} a leap year? {result}") </pre> <p>PROBLEMS ③ OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <p>conda : The term 'conda' is not recognized as the name of a cmdlet, function, script or topic. At line:1 char:1 + conda activate Shashidhar + ~~~~~ + CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException + FullyQualifiedErrorId : CommandNotFoundException</p> <ul style="list-style-type: none"> <li>● PS C:\Users\shash\Downloads&gt; &amp; "c:\Users\shash\anaconda3\envs\Shashidhar\python.exe" Is 2000 a leap year? True</li> <li>● PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\Downloads\AAC A(4.3).py'</li> <li>○ Is 1900 a leap year? False</li> <li>○ PS C:\Users\shash\Downloads&gt; </li> </ul>	
	<p><b>Task 2: One-Shot Prompting – Centimeters to Inches Conversion</b></p>	

	<p><b>Scenario</b> One-shot prompting guides AI using a single example.</p> <p><b>Task Description</b> Use one-shot prompting by providing one input-output example to generate a Python function that:</p> <ul style="list-style-type: none"> <li>• Converts centimeters to inches</li> <li>• Uses the correct mathematical formula</li> </ul> <p><b>Example provided in prompt:</b> Input: 10 cm → Output: 3.94 inches</p> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>• Python function with correct conversion logic</li> <li>• Accurate calculation</li> <li>• Sample test cases and outputs</li> </ul>  <pre> 1 # lab - 2.css      JS lab - 2.js      resume devops.html      # resume dev 2 C: &gt; Users &gt; shash &gt; Downloads &gt; AAC A(4.3).py ... 3   1 def cm_to_inches(cm): 4     2     Convert centimeters to inches. 5     3 6     4     Args: 7     5         cm (float): Length in centimeters. 8     6 9     7     Returns: 10    8         float: Length in inches, rounded to 2 decimal places. 11    9     """ 12   10         inches = cm * 0.393701 13   11         return round(inches, 2) 14 15   12 # Test the function (based on example) 16   13 test_cm = 10 17   14 result = cm_to_inches(test_cm) 18   15 print(f"{test_cm} cm = {result} inches") 19 20   16 # Additional test 21   17 test_cm2 = 5.5 22   18 result2 = cm_to_inches(test_cm2) 23   19 print(f"{test_cm2} cm = {result2} inches") </pre> <p>PROBLEMS 3    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS</p> <ul style="list-style-type: none"> <li>PS C:\Users\shash\Downloads&gt; c;; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\Downloads\AAC A(4.3).py' Is 1900 a leap year? False</li> <li>PS C:\Users\shash\Downloads&gt; c;; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\Downloads\AAC A(4.3).py' 10 cm = 3.94 inches 25 cm = 9.84 inches</li> <li>PS C:\Users\shash\Downloads&gt; c;; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\Downloads\AAC A(4.3).py' 10 cm = 3.94 inches 5.5 cm = 2.17 inches</li> <li>PS C:\Users\shash\Downloads&gt; [ ]</li> </ul>	
<b>Task 3: Few-Shot Prompting – Name Formatting</b>		

	<p><b>Scenario</b>  Few-shot prompting improves accuracy by providing multiple examples.</p> <p><b>Task Description</b>  Use few-shot prompting with 2–3 examples to generate a Python function that:</p> <ul style="list-style-type: none"> <li>• Accepts a full name as input</li> <li>• Formats it as “Last, First”</li> </ul> <p><b>Example formats:</b></p> <ul style="list-style-type: none"> <li>• "John Smith" → "Smith, John"</li> <li>• "Anita Rao" → "Rao, Anita"</li> </ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>• Well-structured Python function</li> <li>• Output strictly following example patterns</li> <li>• Correct handling of names</li> <li>• Sample inputs and outputs</li> </ul>  <pre data-bbox="453 620 1209 1712"> # lab - 2.css      JS lab - 2.js      resume devops.html      # resume devops.css      JS res C: &gt; Users &gt; shash &gt; Downloads &gt; AAC A(4.3).py &gt; ... 1 def format_name(full_name): 2 3     Format a full name from 'First Last' to 'Last, First'. 4 5     Args: 6         full_name (str): The full name as a string. 7 8     Returns: 9         str: Formatted name as 'Last, First'. 10    """ 11    parts = full_name.strip().split() 12    if len(parts) &gt;= 2: 13        first = parts[0] 14        last = parts[-1] # Handles extra middle names if present 15        return f"{last}, {first}" 16    else: 17        return full_name # Fallback for invalid input 18 19 # Test the function 20 test_name = "Shashidhar Ashadapu" 21 result = format_name(test_name) 22 print(f"Formatted: {result}") </pre> <pre data-bbox="518 1474 1209 1712"> PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\anaconda\nloads\AAC A(4.3).py' ● PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\anaconda\nloads\AAC A(4.3).py' 10 cm = 3.94 inches 5.5 cm = 2.17 inches ● PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\anaconda\nloads\AAC A(4.3).py' Formatted: Wonderland, Alice ● PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\Users\shash\anaconda\nloads\AAC A(4.3).py' Formatted: Ashadapu, Shashidhar ● PS C:\Users\shash\Downloads&gt; </pre> <p><b>Task 4: Comparative Analysis – Zero-Shot vs Few-Shot</b></p> <p><b>Scenario</b>  Different prompt strategies may produce different code quality.</p> <p><b>Task Description</b></p> <ul style="list-style-type: none"> <li>• Use zero-shot prompting to generate a function that counts vowels in a string</li> <li>• Use few-shot prompting for the same problem</li> </ul>	
--	--	--

- 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

The screenshot shows a VS Code interface with a terminal window open. The terminal displays the following Python code:

```

C:\> Users > shash > Downloads > AAC A(4.3).py > ...
1 def count_vowels(text):
2     """
3         Count the number of vowels in a given text.
4
5     Args:
6         text (str): The input string.
7
8     Returns:
9         int: Number of vowels (a, e, i, o, u, case-insensitive).
10    """
11    vowels = 'aeiouAEIOU'
12    count = 0
13    for char in text:
14        if char in vowels:
15            count += 1
16    return count
17
18 # Test
19 test_text = "Shashidhar Ashadapu"
20 result = count_vowels(test_text)
21 print(f"Vowels in '{test_text}': {result}")

```

Below the code, the terminal shows the output of running the script:

```

xe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32\lib\site-packages\debugpy\launcher' '50181' '--' 'c:\Users\shash\Downloads\AAC A(4.3).py'
Formatted: Ashadapu, Shashidhar
PS C:\Users\shash\Downloads> c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32\lib\site-packages\debugpy\launcher' '50181' '--' 'c:\Users\shash\Downloads\AAC A(4.3).py'
Vowels in 'hello world': 3
PS C:\Users\shash\Downloads> c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32\lib\site-packages\debugpy\launcher' '50890' '--' 'c:\Users\shash\Downloads\AAC A(4.3).py'
Vowels in 'Shashidhar Ashadapu': 7
PS C:\Users\shash\Downloads> []

```

**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

	os.html	# resume devops.css	JS resume devops.js	↳ Tavinos.html	# Tavinos.js	
<pre>C: &gt; Users &gt; shash &gt; Downloads &gt; AAC A(4.3).py &gt; ... 1  def count_lines_in_file(file_path): 2      """ 3          Count the number of lines in a text file. 4      """ 5      Args: 6          file_path (str): Path to the .txt file. 7 8      Returns: 9          int: Number of lines, or 0 if file not found. 10 11     Raises: 12         FileNotFoundError: If the file doesn't exist. 13     """ 14     try: 15         with open(file_path, 'r') as file: 16             lines = file.readlines() 17             return len(lines) 18     except FileNotFoundError: 19         print(f"Error: File '{file_path}' not found.") 20     return 0 21 22 # Create a sample file for testing (run this once) 23 with open('sample.txt', 'w') as f: 24     f.write("Line1\nLine2\nLine3\nLine4\nLine5") 25 26 # Test the function 27 result = count_lines_in_file('sample.txt') 28 print(f"Line count in 'sample.txt': {result}")  </pre> <p>PROBLEMS 3    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS</p> <pre>undled\libs\debugpy\launcher' '50181' '--' 'c:\Users\shash\Downloads\AAC Vowels in 'hello world': 3 PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\U undled\libs\debugpy\launcher' '50890' '--' 'c:\Users\shash\Downloads\AAC Vowels in 'Shashidhar_Ashadapu': 7 ● PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\U undled\libs\debugpy\launcher' '61017' '--' 'c:\Users\shash\Downloads\AAC Line count in 'sample.txt': 3 ● PS C:\Users\shash\Downloads&gt; c:; cd 'c:\Users\shash\Downloads'; &amp; 'c:\U undled\libs\debugpy\launcher' '64519' '--' 'c:\Users\shash\Downloads\AAC Line count in 'sample.txt': 5 ○ PS C:\Users\shash\Downloads&gt;</pre>						

**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.