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) NS2 (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**	24CSBTB01 To 24CSBTB39
Assignment Number:3.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab Experiment: Prompt Engineering – Improving Prompts and Context Management (0.5 marks)**	Week2 - Monday

Objective

To explore how prompt design and context influence AI-generated outputs and to learn techniques to improve AI responses.

Tools Required

- GitHub Copilot / Google Gemini / ChatGPT
- VS Code / Google Colab
- Internet access

Procedure

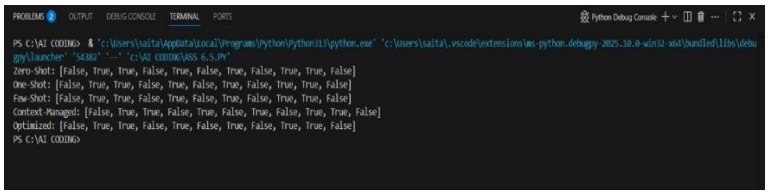
1. Select a simple task: *"Write a Python function to check if a number is prime."*
2. Use different prompting strategies to generate the solution:
    a) Zero-Shot – no examples.
    b) One-Shot – one example provided.
    c) Few-Shot – multiple examples provided.
    d) Context-Managed – detailed prompt with constraints and instructions.
3. Record AI responses and refine prompts to improve code quality.
4. Request AI to optimize the logic for efficiency.
5. Compare results and document improvements.

CODE :-

```python
# Zero-Shot Prompt: Write a Python function to check if a number is prime.
def is_prime_zero_shot(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

# One-Shot Prompt: Example - is_prime(5) should return True.
def is_prime_one_shot(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

# Few-Shot Prompt: Examples - is_prime(2) -> True, is_prime(4) -> False, is_prime(7) -> True.
def is_prime_few_shot(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
```

```python
# Context-Managed Prompt: Constraints - efficient, handle edge cases, use minimal loops, clear variable names.
def is_prime_context_managed(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for divisor in range(3, int(n ** 0.5) + 1, 2):
        if n % divisor == 0:
            return False
    return True

# Optimized Logic: Use square root, skip even numbers, handle small numbers efficiently.
def is_prime_optimized(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

# Test cases to compare results
if __name__ == "__main__":
    test_numbers = [1, 2, 3, 4, 5, 9, 11, 25, 29, 97, 100]
    print("Zero-Shot:", [is_prime_zero_shot(x) for x in test_numbers])
    print("One-Shot:", [is_prime_one_shot(x) for x in test_numbers])
    print("Few-Shot:", [is_prime_few_shot(x) for x in test_numbers])
    print("Context-Managed:", [is_prime_context_managed(x) for x in test_numbers])
    print("Optimized:", [is_prime_optimized(x) for x in test_numbers])
```

OUTPUT :-



Sample Prompts

	• Zero-Shot: Write a Python function to check if a number is prime. • One-Shot: Example: Input: 5 → Output: Prime. Now, write a function to check if a number is prime. • Few-Shot: Example 1: Input: 7 → Output: Prime Example 2: Input: 10 → Output: Not Prime Example 3: Input: 2 → Output: Prime Generate the function accordingly. • Context-Managed (With Optimization)	
2	**Task: Mobile Data Usage Billing Application (1.0 Marks)** **Objective:** Use Python programming and AI-assisted coding tools to create an application that simulates mobile data billing for a telecom service provider. Instructions 1. Use GitHub Copilot or Google Gemini to assist in writing the program. 2. Read the following inputs from the user: o Data Consumed (in GB) o Plan Type (Prepaid / Postpaid) o Additional Services Used (e.g., caller tune, OTT subscription, etc.) 3. Implement billing logic to calculate: o DC (Data Charges) – charges based on data consumption o VC (Value-added Charges) – charges for additional services o Tax – applicable tax on the total bill	Week2 - Monday

4. Display an itemized bill showing:
    - Plan Type
    - Data Usage and Charges
    - Value-added Services and Charges
    - Tax
    - Total Bill Amount

Requirements

- Students must refer to their actual mobile bill for charge structure (data cost, service fees, taxes) to make the program realistic.
- AI assistance (Copilot/Gemini) must be used to generate and refine the initial code.

Deliverables

- AI prompts used for code generation.
- AI-generated Python code and any optimized version.
- Screenshots of:
    - AI interactions
    - Program execution and output
    - Comparison with the student's actual mobile bill.

CODE :-

```python
def calculate_bill(data_gb, plan_type, services):
    # Rates (sample realistic values – update from your own bill)
    data_rate = 10 if plan_type.lower() == "prepaid" else 8  # ₹ per GB
    service_rates = {
        "caller tune": 30,
        "ott": 150,
        "international roaming": 500
    }
    tax_rate = 0.18  # 18% GST

    # Calculate charges
    data_charges = data_gb * data_rate
    value_added_charges = sum(service_rates.get(s.lower(), 0) for s in services)
    subtotal = data_charges + value_added_charges
    tax = subtotal * tax_rate
    total = subtotal + tax

    # Display bill
    print("\n--- Mobile Billing Summary ---")
    print(f"Plan Type: {plan_type}")
    print(f"Data Used: {data_gb} GB @ ₹{data_rate}/GB = ₹{data_charges:.2f}")
    print(f"Value-Added Services: {', '.join(services)} = ₹{value_added_charges:.2f}")
    print(f"Tax (18% GST): ₹{tax:.2f}")
    print(f"Total Amount Payable: ₹{total:.2f}")
```

```python
# ---- Input ----
data_gb = float(input("Enter data used (in GB): "))
plan_type = input("Enter plan type (Prepaid/Postpaid): ")
services_input = input("Enter additional services used (comma-separated): ")
services = [s.strip() for s in services_input.split(",") if s.strip()]

# ---- Billing ----
calculate_bill(data_gb, plan_type, services)
```

OUTPUT:-

```
PS C:\AI CODEING>  & 'c:\Users\kbhuv\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\k
 '59463' '--' 'c:\AI CODEING\lab ass 1.1.py'
Enter data used (in GB): 5
Enter plan type (Prepaid/Postpaid): postpaid
Enter additional services used (comma-separated): comma-separated

--- Mobile Billing Summary ---
Plan Type: postpaid
Data Used: 5.0 GB @ ₹8/GB = ₹40.00
Value-Added Services: comma-separated = ₹0.00
Tax (18% GST): ₹7.20
Total Amount Payable: ₹47.20
```

3	**Task: Develop an LPG Billing System (1.0 Marks)** **Objective** Apply your Python programming skills and utilize AI-assisted coding tools to build an application that calculates the LPG bill based on specified customer inputs and billing parameters. Instructions 1. Use GitHub Copilot or Google Gemini to assist in writing and refining the program. 2. Read the following user inputs: o Cylinder Type (Domestic 14.2 kg / Domestic 5 kg / Commercial 19 kg / Commercial 47.5 kg) o Number of Cylinders Booked o Subsidy Amount (applicable only for domestic cylinders) 3. Refer to the given LPG Price List to determine the price per cylinder: o Domestic LPG (14.2 kg) → ₹905.00 o Domestic LPG (5 kg) → ₹335.50	Week2 - Monday

- Commercial LPG (19 kg) → ₹1,886.50
- Commercial LPG (47.5 kg) → ₹4,712.00
- Delivery Charges (₹10 to ₹50)

4. Implement the billing formula:

Bill Amount = (Price per Cylinder × Quantity) - Subsidy (if applicable) + Delivery Charges

5. Calculate and display an itemized bill including:

- Cylinder Type
- Number of Cylinders
- Base Amount
- Subsidy
- Delivery Charges
- Total Bill Amount

Deliverables

- A report containing:
  - AI prompts used to generate the program
  - AI-generated Python code
  - Line-by-line explanation of the code

CODE :-

```python
def calculate_lpg_bill(cylinder_type, quantity, subsidy, delivery_charge):
    # Price list
    prices = {
        "Domestic 14.2 kg": 905.00,
        "Domestic 5 kg": 335.50,
        "Commercial 19 kg": 1886.50,
        "Commercial 47.5 kg": 4712.00
    }

    # Determine base amount
    price_per_cylinder = prices.get(cylinder_type, 0)
    base_amount = price_per_cylinder * quantity

    # Apply subsidy only for domestic
    subsidy_amount = subsidy if "Domestic" in cylinder_type else 0

    # Final bill
    total = base_amount - subsidy_amount + delivery_charge

    # Display itemized bill
    print("\n--- LPG BILL ---")
    print(f"Cylinder Type      : {cylinder_type}")
    print(f"Number of Cylinders: {quantity}")
    print(f"Base Amount        : ₹{base_amount:.2f}")
    print(f"Subsidy            : ₹{subsidy_amount:.2f}")
    print(f"Delivery Charges   : ₹{delivery_charge:.2f}")
```

```
    print(f"Total Bill Amount  : ₹{total:.2f}")

# --- Input section ---
ctype = input("Enter Cylinder Type: ")
qty = int(input("Enter Number of Cylinders: "))
subs = float(input("Enter Subsidy Amount (₹): "))
delivery = float(input("Enter Delivery Charges (₹10–₹50): "))

# --- Billing function call ---
calculate_lpg_bill(ctype, qty, subs, delivery)
```

OUTPUT :-

```
PS C:\AI CODEING> & c:\Users\kbhuv\AppData\Local\Pr
  '62851' '--' 'c:\AI CODEING\lab ass 1.1.py'
Enter Cylinder Type: hp
Enter Number of Cylinders: 3
Enter Subsidy Amount (₹): 800
Enter Delivery Charges (₹10–₹50): 10

--- LPG BILL ---
Cylinder Type      : hp
Number of Cylinders: 3
Base Amount        : ₹0.00
Subsidy            : ₹0.00
Delivery Charges   : ₹10.00
Total Bill Amount  : ₹10.00
```