Lab Assignment-3.5

Prompt Engineering — Improving Prompts and Context Management

V.SRICHARAN

2303A510G0

Batch-Xii

Task-I: "Write a Python function to check if a number is prime."

Prompts used for best results:

#Write a python code for checking a prime number.

#0ptimize the logic for better performance

Code:

```python
#Write a python code for checking a prime number.
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
input_number = int(input("Enter a number: "))
if is_prime(input_number):
    print(f"{input_number} is a prime number.")
else:
    print(f"{input_number} is not a prime number.")

```

```python
#Optimize the logic for better performance
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
input_number = int(input("Enter a number: "))
if is_prime_optimized(input_number):
    print(f"{input_number} is a prime number.")
else:
    print(f"{input_number} is not a prime number.")

```

Output:

```
/Users/chari/OneDrive/Desktop/AI Assis

Enter a number: 741852963
741852963 is not a prime number.
```

Comparision of both results: The standard is_prime function checks all numbers up to the square root of n, making it simple but less efficient for large numbers. The optimized version, is_prime optimized, quickly eliminates even numbers and multiples of 3, then only checks numbers of the form 61<+1, reducing unnecessary checks. Both have O(Nn) complexity, but the optimized version is faster for large inputs. Use is_prime for clarity and is_prime optimized for better performance with big numbers.

Task-2: Mobile Data Usage Billing Application

Prompts used:

Use Python programming and AI-assisted coding tools to create an application that simulates mobile data billing for a telecom service provider.

Generate python code such that user should have Data Consumed (in GB), Plan Type (Prepaid / Postpaid),Additional Services Used (e.g., caller tune, OTT subscription, etc.)

#Implement billing logic to calculate: DC (Data Charges) — charges based on data consumption, VC (Value-added Charges) — charges for additional services, Tax applicable tax on the total bill.
#Display an itemized bill showing: Plan Type, Data Usage and Charges, Value-added Services and Charges, Tax, Total Bill Amount

Code:

```
345 class MobileDataP1anItemized:
346 def init (self, plan_name, data_limit_gb, cost_per_gb, service cost, tax_rate) :
347 self. plan_name = plan_name 348 self. = data_limit_gb 349 self. cost_per_gb = cost_per_gb
350 self. service cost = service cost 351 self. tax rate = tax rate
352
353 def calculate bill (self, :
354 if data_used_gb self. data_limit_gb:
355 data charges = data_used_gb * self. cost_per_gb 356 else:
357 extra_data = data used_gb - self. data limit_gb 358 data_charges = (self.data_limit_gb * self.cost_per_gb) + (extra_data *
self. cost_per_gb * 1.5) # 50% surcharge 359 total before_tax = data charges + self. service_cost
360 tax amount = total before tax * self. tax rate 361 total bill = total before tax + tax amount
362 return data_charges, self. service_cost, tax_amount, total bill

364 def display_itemized_bill(self, data_used_gb, plan_type, value_added_services) :
365 data charges, service_charges, tax amount, total bill = self. calculate bill(data_used_gb) 366
print("\n--- Itemized Mobile Data Bill 367 print(f"P1an Type: {plan_type}")
368 print (f" Plan Name: {self. plan_name}" )
369 print(f"Data Usage: {data used_gb} 6B") 370 print (f "Data Charges: Rs.{data_charges: .2f}") 371
print (f "Value-added Services: { . join(value added services) if value added services else None' Y') 372
print(f"Va1ue-added Services Charges: Rs.{service_charges: .2f}")
373 print(f"Tax: Rs . {tax_amount: .2f}")
374 print(f"Tota1 Bill Amount: Rs.{total_bill: .2f}")
375
376 # Define available plans
377 basic_plan = MobileDataP1anItemized("Basic", 5, lø, 50, e. 18) # 5 6B, Rs.10/GB, Rs.50 services, 18% tax
378 premium_plan = 20, 8, lee, 0.18) # 20 6B, Rs.8/GB, Rs.10Ø services, 18% tax
379
380 # User input
381 data_used = float(input("Enter data used in 6B:
382 plan_type = input("Se1ect plan type (Prepaid/ Postpaid) : 383 selected_plan =
input("Se1ect plan (Basic/ Premium) :
```

```
# Value-added services services — add services = input("Did you use any value-added
services? (yes/no) : . strip() . lower() if add services = "yes " • while True:
    service = input("Enter service name (or press Enter to finish): . strip() if
    service:
        services. append
    (service) else:
        break

# Bill calculation and display if
selected_plan. lower() = "basic" .
    basic_plan.display_itemized bill (data used, plan_type, services) elif
selected_plan.lower() = premium" :
    premium_plan.display_itemized bill (data used, plan_type, services) else:
    print("Invalid plan selected. ")
```

Output:

```
Enter data used in GB: 3
Select plan type (Prepaid/Postpaid): postpaid
Select plan (Basic/Premium): premium
Plan Type: Premium
Data Usage: 3.0 GB
Data Charges: Rs.24.00
Value-added Services Charges: Rs.100.00
Tax: Rs.22.32
Total Bill Amount: Rs.146.32
PS C:\Users\chari>
```

Comparision of both results: This Mobile Data Usage Billing Application helps users calculate their monthly mobile data bill in a simple and interactive way. The user selects their plan type (Prepaid or Postpaid) and chooses between a Basic or Premium plan, each with its own data limits and rates. The app asks for the amount of data used and whether any valueadded services (like caller tunes or OTT subscriptions) were used. It then calculates the total bill, including extra charges for exceeding the data limit, service costs, and applicable taxes. Finally, it presents a clear, itemized bill that breaks down all charges, making it easy for users to understand exactly what they're paying for. This makes managing and reviewing mobile expenses straightforward and transparent.

Task-3: Develop an LPG Billing System

Prompts used:

#Develop a Python application and utilize Al-assisted coding tools to build an application that calculates the LPG bill based on specified customer inputs and billing parameters.

#Generate python code such that user should have Customer Name, Customer ID, Consumption (in kg), Connection Type (Domestic / Commercial)

Refer to the given LPG Price List to determine the price per cylinder:

Add delivery charge input and detailed billing class with display method

Calculate per kg price from cylinder price

Code:

```
346    def calculate bill (self, consumption_kg) :
347    gross amount = consumption_kg * self. base_price_per_kg 348 subsidy_amount =
       consumption kg * self.
```

```
349 net amount = gross amount - subsidy_amount 350 tax amount = net amount * self. tax rate 351 total bill = net amount + tax amount
352 return gross amount, subsidy amount, net amount, tax amount, total bill 353 def init (self, customer name, customer id, consumption_kg, connection_type) :
354 self. customer name = customer name 355 self. customer id = customer id 356 self. consumption_kg = consumption kg
357 self. connection_type = connection_type 358
359 def display_customer_info (self) :
360 print(f"Customer Name: {self. customer name}" )
361 print(f"Customer ID: {self. customer_id}")
362 print(f"Consumption: {self. consumption_kg} kg")
363 print(f"Connection Type: {self. connection_type}" )
364 def
365 while True:
366 try:
367 charge = float(input("Enter Delivery Charge (<10 to <5Ø) :
368 if 10 <= charge <= 50: 369 return charge 370
else:
371 print("De1ivery charge must be between and *50.
372 except ValueError:
373 print("P1ease enter a valid number.
374
375 class LPGBi11ingSystemDetai1ed:
376 def init (self, base_price_per_kg, subsidy_per_kg, tax rate, delivery_charge=e) :
377 self. = base_price_per_kg
378 self. subsidy_per_kg = subsidy_per_kg
379 self. tax rate = tax rate
380 self. delivery_charge = delivery_charge
381
382 def calculate bill (self, consumption_kg) :
383 gross amount = consumption kg * self.
384 subsidy amount = consumption_kg * self.
385 net_amount = gross_amount - subsidy_amount 386 tax amount = net amount * self. tax rate
390 def display_detailed_bill(self, customer, consumption_kg) :
391 gross, subsidy, net, tax, delivery, total = self. calculate bill(consumption_kg) 392 print("\nltemized Bill:
393 customer. display_customer_info( ) 394print(f"Gross Amount: Rs.{gross: .2f}") 395 print(f"Subsidy Amount: Rs.{subsidy: .2f}") 396 print(f"Net Amount: Rs. {net: .2f}") 397 print(f"Tax Amount: Rs. {tax: .2f}") 398 print(f"De1ivery Charge: Rs. {delivery: .2f}") 399 print(f"Tota1 Bill Amount: Rs.{total: .2f}")
400
401 # Main billing logic
402 cylinder_size = float(input("Enter Cylinder Size in kg (5, 14.2, 19, 47.5): 403 price_per_cylinder = cylinder_size) 404 v if price_per_cylinder is None:
405 print("lnvalid connection type or cylinder size. ") 406 exit ( )
407 v else:
408 print(f"Price per cylinder for {cylinder_size} kg LPG: Rs. :
49 q
```

```
410   base_price_per_kg = price_per_cylinder / cylinder_size
411   subsidy_per_kg = 10 if connection_type.lower() == "domestic" else 0
412   tax_rate = 0.05 if connection_type.lower() == "domestic" else 0.18
413   delivery_charge = get_delivery_charge()
414
415   detailed_billing_system = LPGBillingSystemDetailed(base_price_per_kg, subsidy_per_kg, tax_rate, delivery_charge)
416   detailed_billing_system.display_detailed_bill(customer, consumption)
```

Output:

```
Itemized Bill:
Customer Name: hafgh
Customer ID: 234
Consumption: 5.0 kg
Connection Type: Domestic
Gross Amount: Rs.500.00
Subsidy Amount: Rs.100.00
Net Amount: Rs.400.00
Tax Amount: Rs.20.00
Total Bill Amount: Rs.420.00
```

Comparision of both results: This LPG Gas Billing Application makes it easy for customers to calculate their monthly gas bill. Users enter their personal details, connection type (Domestic or Commercial), cylinder size, and the amount of gas consumed. The app automatically applies the correct price per cylinder, calculates any government subsidy, adds delivery charges, and computes the applicable tax. It then presents a clear, itemized bill showing all charges, including gross amount, subsidy, net amount, tax, and delivery fees. This helps users understand exactly what they're paying for and ensures transparency in their LPG billing. The process is straightforward, making it simple for anyone to review and manage their household or business gas expenses.