Vending Machine Design-Low Level

Vending machines have become an essential part of our everyday lives, offering various kinds of products starting from snacks and beverages to personal care items. While their capability can also appear simple from a user perspective, the low-level design of a vending machine includes complex info to ensure clean

1. Requirements Gathering for Vending Machine


```
class UserInterface:
```

```
    def __init__(self, inventory, payment_processor):
        self.inventory = inventory
        self.payment_processor = payment_processor

    def display_products(self):
        for product in self.inventory.get_products():
            print(f"{product.id}: {product.name} - ${product.price}")

    def select_product(self, product_id):
        product = self.inventory.get_product_by_id(product_id)
        if product:
            return product
        else:
            print("Product not available.")
            return None

    def process_payment(self, product):
        amount = float(input("Insert cash amount: "))
        if self.payment_processor.validate_payment(product.price,
amount):
            self.payment_processor.complete_transaction(product.price)
            self.inventory.update_stock(product)
            print("Transaction successful. Dispensing product...")
            return True
        else:
            print("Insufficient funds or invalid payment.")
            return False
class PaymentProcessor:
    def __init__(self):
        self.current_balance = 0.0

    def validate_payment(self, price, amount):
        return amount >= price

    def complete_transaction(self, amount):
        self.current_balance += amount

    def get_balance(self):
        return self.current_balance
```

```python
class Product:
    def __init__(self, product_id, name, price, quantity):
        self.id = product_id
        self.name = name
        self.price = price
        self.quantity = quantity

class InventoryManagement:
    def __init__(self):
        self.products = []

    def add_product(self, product):
        self.products.append(product)

    def get_products(self):
        return self.products

    def get_product_by_id(self, product_id):
        for product in self.products:
            if product.id == product_id and product.quantity > 0:
                return product
        return None

    def update_stock(self, product):
        product.quantity -= 1
```

```python
class DispensingMechanism:
    def dispense(self, product):
        print(f"Dispensing {product.name}")
```

```python
class SecurityMeasures:
    def __init__(self):
        self.security_logs = []

    def log_transaction(self, transaction):
        self.security_logs.append(transaction)
        print("Transaction logged for security.")
```

```
class VendingMachine:
    def __init__(self):
        self.inventory = InventoryManagement()
        self.payment_processor = PaymentProcessor()
        self.user_interface = UserInterface(self.inventory,
self.payment_processor)
        self.dispensing_mechanism = DispensingMechanism()
        self.security_measures = SecurityMeasures()

    def add_product_to_inventory(self, product):
        self.inventory.add_product(product)

    def start(self):
        while True:
            self.user_interface.display_products()
            product_id = int(input("Select product ID: "))
            product = self.user_interface.select_product(product_id)
            if product:
                if self.user_interface.process_payment(product):
                    self.dispensing_mechanism.dispense(product)
                    self.security_measures.log_transaction(f"Dispensed
{product.name} for ${product.price}")
                else:
                    print("Transaction failed.")
            else:
                print("Invalid product selection.")

if __name__ == "__main__":
    vm = VendingMachine()
    vm.add_product_to_inventory(Product(1, "Coke", 1.50, 10))
    vm.add_product_to_inventory(Product(2, "Pepsi", 1.50, 10))
    vm.add_product_to_inventory(Product(3, "Water", 1.00, 20))
    vm.start()
```

Output ;

```
1: Coke - $1.5
2: Pepsi - $1.5
3: Water - $1.0
Select product ID: 3
```

```
Insert cash amount: 120
Transaction successful. Dispensing product...
Dispensing Water
Transaction logged for security.
1: Coke - $1.5
2: Pepsi - $1.5
3: Water - $1.0
Select product ID: 3
Insert cash amount: 12
Transaction successful. Dispensing product...
Dispensing Water
Transaction logged for security.
1: Coke - $1.5
2: Pepsi - $1.5
3: Water - $1.0

Select product ID:
```