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 operation, robustness, and safety.

1. Requirements Gathering for Vending Machine

1.1 Functional Requirements for the Vending Machine

Functional Requirements generally describe and define features of the end product of a software system and simply focus on what the end product does. These are the requirements that a system should accomplish or do like calculations, data manipulations. Functional requirements of the Vending Machine are:

The vending machine has to maintain inventory records.
The machine should allow a user to select an item and insert cash.
When cash is inserted, the machine should verify that it matches the amount of the chosen item.
When an item is unavailable or there is insufficient cash, the machine must display an error.
Upon successful verification and transaction, the vending machine shall dispense the chosen item to the user.
Ultimately, the user receives the chosen item if all of the previously mentioned stages are successful.

1.2 Non-Functional Requirements for the Vending Machine

Non-functional Requirements in software engineering refer to the characteristics of a software system that are not related to specific functionality or behaviour. They describe how the system should perform, rather than what it should do. Non-Functional requirements of the Vending Machine are:

The inventory records and cash verification mechanisms shall operate with high accuracy to ensure correct transactions.
The vending machine shall provide timely responses, displaying errors promptly and dispensing items efficiently.
The user interface shall be designed to be intuitive, making it easy for users to select items, insert cash, and understand error messages.
The vending machine shall be reliable, minimising downtime and ensuring consistent functionality.
The system shall incorporate security measures to prevent unauthorised access to inventory records and cash transactions.
The vending machine shall be available for use during specified operational hours, minimising any periods of inactivity for maintenance or other reasons.

2. High-Level Design (HLD) of Vending Machine

In the High-Level Design section, the general structure and essential additives of the vending machine are outlined.

How people will use the machine, how money transactions will work, how the products will be managed, and how the machine will talk to other devices ie communication protocols.
Key selections, including the combination of a coin mechanism, bill acceptor, card reader, and display unit, are made at this stage.
The HLD acts as a blueprint for the following Low-Level Design (LLD) section, guiding the particular implementation of everything.

3. Low-Level Design (LLD) of Vending Machine

vending-machine-lld
The Low-Level Design (LLD) is a crucial segment in the development of a vending machine system, following the High-Level Design (HLD).

In this segment, the focal point shifts from the conceptual structure to a more unique and granular level of implementation.
The LLD digs into the specifics of every aspect, both hardware and software, presenting a roadmap for builders to translate the high-level concepts into a functioning device.
It acts as a bridge among the summary illustration of the device inside the HLD and the unique implementation that happens inside the coding phase.

4. Components of Vending Machine with respect to HLD

Building upon the High-Level Design (HLD), the Low-Level Design (LLD) explains each primary element, outlining how they make a contribution to the general capability of the vending machine system. For example:

User Interface Logic:
In the LLD, the user interface logic is broken all the way down to specify how the keypad or touchscreen inputs are processed.
It details how the display unit is controlled to show off product information and transaction status.
Payment Processing:
This component explains how the machine checks if coins are valid, how it handles bills, and how it processes card transactions.
It outlines the communication protocols among the software and the payment hardware components.
Inventory Management:
The LLD breaks down the inventory control machine, explaining how the product database is dependent and the way it is updated with sales and restocking activities.
Dispensing Logic:
The detailed plan (LLD) gives clear reasons for how the conveyor machine works.
It also explains how the computer program controls the process of giving out products, making sure it happens in a specific and reliable way.
Security Measures:
LLD covers how tamper detection mechanisms (Tamper detection is the ability of a device to sense that an active attempt to compromise the device integrity , the detection of the threat may enable the device to initiate appropriate defensive actions) and encryption protocols are implemented to secure the vending machine.

5. Components of Vending Machine Low Level Design

The Low-Level Design is going beyond the HLD by way of presenting an exhaustive breakdown of every element:

Coin Mechanism:
Details how the coin mechanism interacts with the software program, which include validation and processing of inserted coins.
Bill Acceptor:
Describes the conversation and validation method involved in accepting paper currency.
Card Reader:
Explains how the card reader interfaces with the software program to authorise credit score or debit card transactions.
Conveyor System and Spiral Mechanism:
Specifies the control mechanisms for transporting merchandise and rotating the spiral for dispensing (a spiral mechanism refers to a specific design element used for dispensing products.
It typically involves a spiral-shaped structure, often made of metal or another durable material, that rotates to move items from their storage location to the dispensing point).
User Interface:
Breaks down the input handling and shows control logic for user interaction.

6. How are classes interacting with each other?

In the Low-Level Design (LLD) of the vending machine system, classes are used to model numerous software entities and their interactions. Here's an in depth clarification of the way classes interact with each other:

User Interface Class:
Communicates with the Payment Processing class to initiate and process transactions based totally on user inputs.
Sends requests to the Inventory Management class to update product data and availability on the display.
Payment Processing Class:
Communicates with the Coin Mechanism class to validate and process inserted cash.
Interfaces with the Bill Acceptor class to authenticate and process paper currency.
Connects with the Card Reader class to authorise credit or debit card transactions.
Notifies the Inventory Management class when it hits successful transactions and update product quantities.
Inventory Management Class:
Receives updates from the Payment Processing class concerning product sales and restocking activities.
Communicates with the Dispensing Logic class to manage product dispensing and stock manage.
Security Measures Class:
Monitors interactions among various classes to hit upon any anomalies or unauthorised access.
Collaborates with the Communication Protocols class to make sure steady information transmission.

Communication Protocols Class:
Facilitates communication between the User Interface, Payment Processing, and Inventory Management class.
Manages serial communication with hardware components together with the Coin Mechanism, Bill Acceptor, and Card Reader.
These interactions (actions or exchanges of information) create a connected flow of data and control between different parts or groups.
This setup makes the design organised into separate and easy-to-manage sections.

7. Principles of Vending Machine Low Level Design(LLD)

The Low-Level Design adheres to several key principles that contribute to the robustness and effectiveness of the Vending machine system:
Modularity:
Components are designed as modular units, allowing for easier maintenance, debugging, and future upgrades.
Each class encapsulates precise capability, promoting a smooth separation of concerns.
Encapsulation:
Classes encapsulate their internal workings, exposing most effective essential interfaces to other classes.
This protects the integrity of each factor and prevents unintended interference.
Abstraction:
Abstraction is employed to simplify the complexities of the system.
Classes provide an excessive-stage evaluation of capability without exposing pointless implementation info.
Communication Cohesion:
Classes communicate with a clean and properly-defined purpose.
This guarantees that each class is accountable for a particular set of tasks, promoting a cohesive and efficient interaction shape.
Information Hiding:
Implementation info within every class is hidden from different classes, reducing dependencies and minimising the impact of changes in one class on others.

8. Conclusion

The detailed plan for a vending machine system needs careful consideration of both its physical parts and the computer programs that run it. Everything from how money is processed to controlling how products are dispensed is crucial to make sure users have a smooth and reliable experience. As technology evolves, adding modern features like touchscreens, card readers, and online connectivity improves the abilities and efficiency of vending machines, making them an important part of our automated retail world.

1. User Interface Class

This class handles user interactions, including selecting products and processing payments.

```python
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
```

2. Payment Processing Class

# This class manages all types of payment methods including cash, bill acceptors, and card readers.

```python
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
```

# 3. Inventory Management Class
This class manages the inventory, including updating stock levels and checking product availability.

```
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
```

4. Dispensing Logic Class
This class is responsible for dispensing products.

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

5. Security Measures Class
This class ensures the security of transactions and inventory.

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

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

6. Main Vending Machine Class
This class integrates all components and provides a cohesive vending machine system.

```python
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.")
```

Running the Vending Machine To use the vending machine, you would create an instance of VendingMachine, add products to the inventory, and start the machine.

```python
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()
```