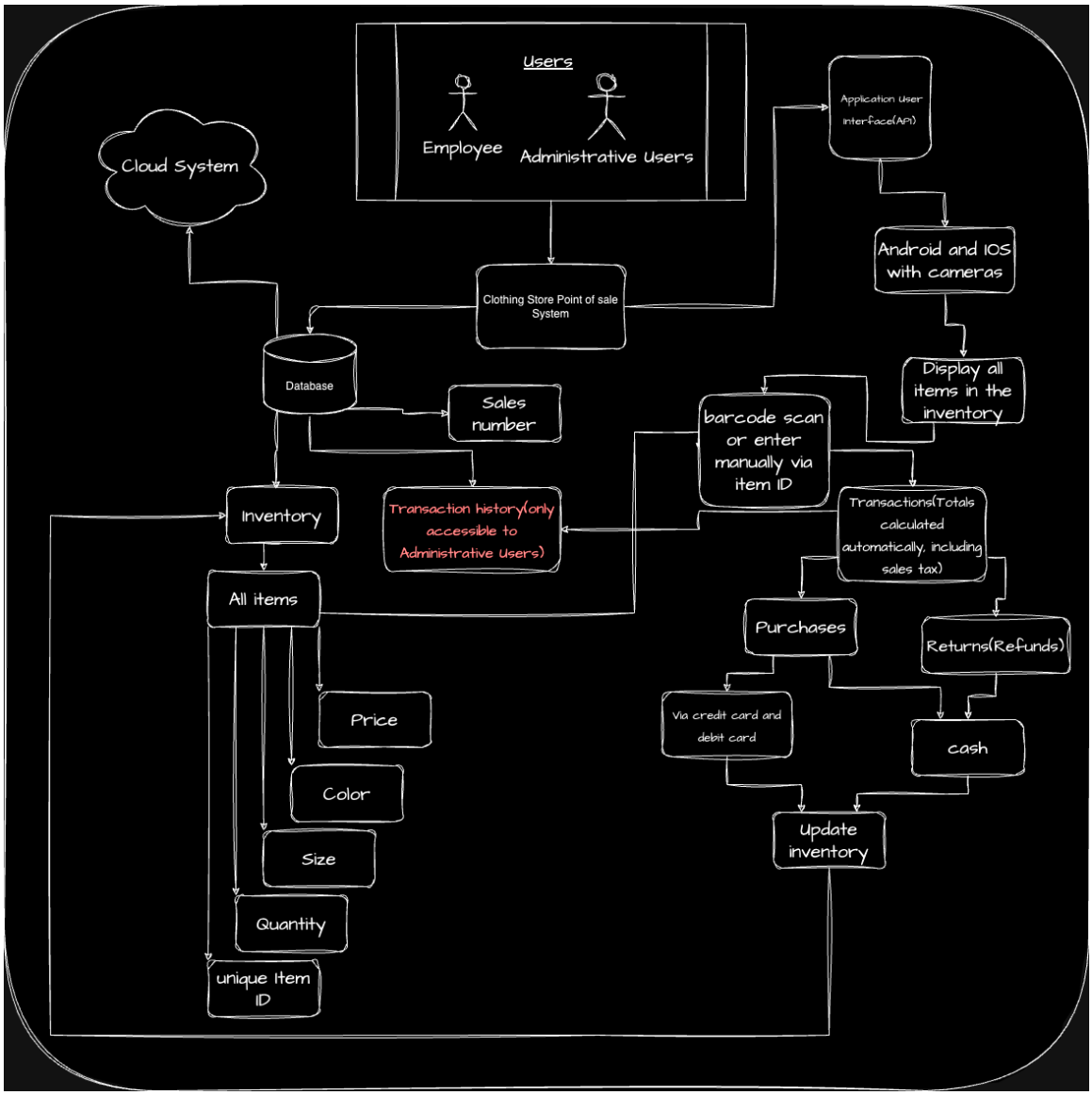
Clothing Store Point of Sale System

Team Members: Adam Bui, Kyle Kucharski, Sanchit Singh

System Description:
The Point of Sale System is designed to make sales efficient and organized for employees, it also helps with managing inventory as well as all kinds of transactions including purchases and returns. The system will allow you to see which items are currently available in stock. This system will help employees in enhancing their productivity and simplify their day to day tasks. By automating various aspects of inventory management, this system aims to improve sales by reducing the workload on employees.

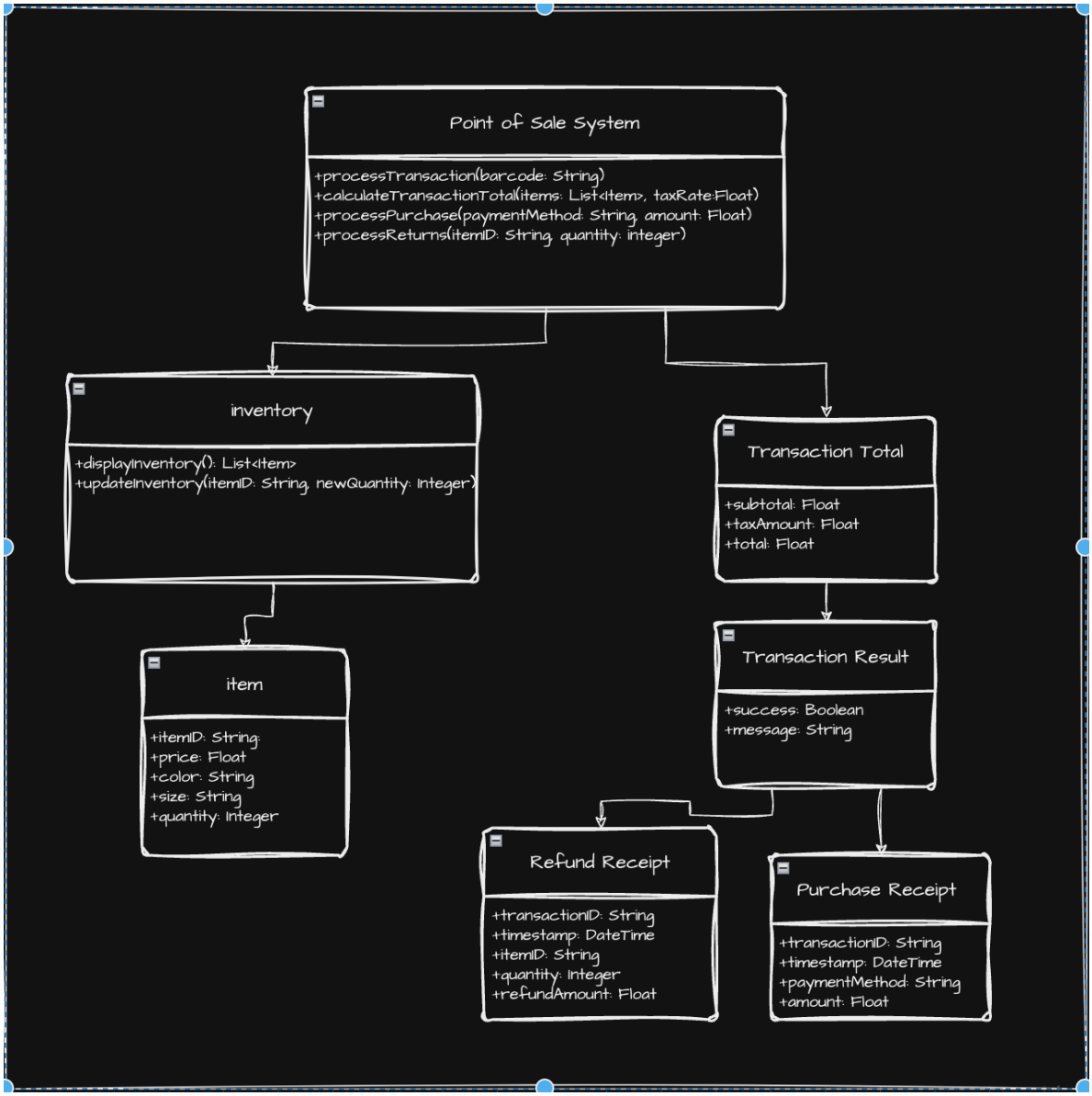
Architectural Diagram:



SWA Description:

The system will have a database that manages the inventory data, sales number, and the transaction history, which will only be accessible to Administrative users. The inventory data will have all the information regarding each item. All items will have a unique item ID assigned to it along with the quantity, color, size, and price. This system will support android and IOS os systems that have cameras. On the device, all the items that are in the inventory will be displayed. To select an item, one can scan the barcode or enter manually via item ID. People can purchase via credit card, debit card, or cash but can only do returns using cash. Once the transaction is completed, the inventory will be updated.

UML Class Diagram:



Description of Classes:

1) **PointOfSaleSystem**

- Purpose: This class represents the core point of sale system and serves as the main entry point for interactions with customers. It orchestrates various operations and manages the transaction process.
- Operations:
    - processTransaction(barcode: String, itemID: String):
        - TransactionResult: This operation processes a transaction, either by scanning a barcode or entering an item ID manually.
    - calculateTransactionTotal(items: List<Item>, taxRate: Float):
        - TransactionTotal: Calculates the total transaction amount, factoring in the price of items and applicable taxes.
    - processPurchase(paymentMethod: String, amount: Float):
        - PurchaseReceipt: Records and processes a purchase, including payment details.
    - processReturn(itemID: String, quantity: Integer):
        - RefundReceipt: Handles product returns, specifying the item and quantity.

2) **Inventory**

- Purpose: This class manages the store's inventory and keeps track of available items.
- Attributes: allItems: List<Item>: A list of all items available in the inventory.
- Operations:
    - displayInventory(): List<Item>: Retrieves and displays the list of all items in the inventory.
    - updateInventory(itemID: String, newQuantity: Integer): Boolean: Updates the inventory by adjusting item quantities based on transactions.

3) **Item**

- Purpose: This class represents individual items available for sale in the store.
- Attributes:
    - itemID: String: A unique identifier for the item.
    - price: Float: The price of the item.
    - color: String: The color of the item.
    - size: String: The size of the item.
    - quantity: Integer: The quantity of the item in stock.

4) **TransactionResult**

- Purpose: This class represents the result of a transaction, providing details about the outcome of the transaction.
- Attributes:
    - success: Boolean: Indicates whether the transaction was successful.
    - message: String`: A message describing the transaction result.

5) TransactionTotal

- Purpose: This class represents the total amount of a transaction, including tax.
- Attributes:
    - subtotal: Float: The subtotal before tax.
    - taxAmount: Float: The amount of tax.
    - total: Float: The final total, including tax.

6) PurchaseReceipt

- Purpose: This class represents a receipt for a purchase transaction.
- Attributes:
    - transactionID: String: A unique identifier for the transaction.
    - timestamp: DateTime: The date and time of the transaction.
    - paymentMethod: String: The payment method used.
    - amount: Float: The total amount paid.

7) RefundReceipt

- Purpose: This class represents a receipt for a product return transaction.
- Attributes:
    - transactionID: String: A unique identifier for the transaction.
    - timestamp: DateTime: The date and time of the transaction.
    - itemID: String: The item being returned.
    - quantity: Integer: The quantity of items returned.
    - refundAmount: Float: The amount refunded to the customer.

Description of Attributes:

1) Cloud System

- It primarily represents the cloud-based infrastructure that supports the system.
- Data Type: CloudSystem (custom data type)
- Variable Name: cloudSystem

2) Database

- Manages inventory data, sales numbers, and transaction history.
- Data Type: DataBase (custom data type)
- Variable Name: dataBase

3) **Sales Number**
- An attribute in the database to keep track of sales numbers.
- Data Type: int
- Variable Name: salesNumber
- Constraints: Non negative values only

4) **Clothing Store Point of Sale**
- It represents the core system component for point-of-sale operations.
- DataType: PointOfSaleSystem (custom data type)
- Variable Name: posSystem

5) **Inventory**
- It represents the store's inventory.
- Data Type: Inventory (string)
- Variable Name: inventory

6) **All Items**
- Represents the entire list of items available in the store's inventory.
- Data Type: List of Items (string)
- Variable Name: allItems

7) **Price:**
- Attribute associated with each item in the inventory, indicating the item's price.
- Data Type: Float
- Variable Name: price
- Constraints: Non negative values only

8) **Color**
- Attribute associated with each item in the inventory, indicating the item's color.
- Data Type: String
- Variable Name: color

9) **Size**
- Attribute associated with each item in the inventory, indicating the item's size.
- Data Type: String
- Variable Name: size

10) **Quantity**

- Attribute associated with each item in the inventory, indicating the quantity of that item in stock.
- Data Type: Int
- Variable Name: quantity
- Constraints: Non-negative values only

11) Unique Item ID
- Attribute associated with each item in the inventory, providing a unique identifier for each item.
- Data Type: String
- Variable Name: itemID
- Constraints: Should be unique for each item

12) Application User
- No specific attributes are mentioned in the diagram. It represents users interacting with the point of sale application.
- Data Type: User (custom data type)
- Variable Name: appUser

13) Android and iOS with Cameras
- Indicates the compatibility of the system with Android and iOS devices equipped with cameras.
- Data Type: MobileDevice (custom data type)
- Variable Name: mobileDevice

Description of Operations:

1) Display All Items in the Inventory
- Method Name: displayInventory
- Input Parameters: None
- Return Type: List of item
- Explanation: Retrieves and displays the list of all items in the inventory

2) Barcode Scan or Enter Manually via Item ID
- Method Name: processTransaction
- Input Parameters: Barcode(string) or ItemID(string)
- Return Type: TransactionResult (custom data type)
- Explanation: Processes a transaction by either scanning a barcode or entering an item ID manually. It then returns the transaction result.

3) Transaction Totals Calculated Automatically, Including Tax

- Method Name: calculateTransactionTotal
- Input Parameters: List of item, TaxRate(float)
- Return Type: TransactionTotal (custom data type)
- Explanation: Calculates the transaction total, including taxes, for a list of items.

4) **Purchases**
- Method Name: processPurchase
- Input Parameters: PaymentMethod(string), Amount(float)
- Return Type: PurchaseReceipt (custom data type)
- Explanation: Records and processes a purchase, specifying the payment method and amount. It also returns a purchase receipt.

5) **Returns**
- Method Name: processReturn
- Input Parameters: ItemID(string), Quantity(int)
- Return Type: RefundReceipt (custom data type)
- Explanation: Handles product returns, specifying the item ID and quantity. It also generates a refund receipt.

6) **Update Inventory**
- Method Name: updateInventory
- Input Parameters: ItemID(string), NewQuantity(int)
- Return Type: Boolean (indicates success or failure)
- Explanation: Updates the inventory by adjusting the quantity of items based on a transaction. It returns a success or failure indicator.

Development Plan and Timeline:

The Clothing Store Point of Sale System should be developed between 3 to 4 months. The following information consists of different team members as well as a recommended timeline of each of the tasks they should be held responsible for completing before the 3 to 4 month period:

1. **Project Manager**
    a. Leads weekly team meetings to give feedback to all teams
    b. Oversees and provides support for the progress of team members
    c. Serves as a liaison between the team and the client to verify and validate the system's functions
2. **Account Security Team**
    a. Ensures system data is being backed up to a cloud every hour
    b. Meets with Test Engineers twice a day to test employee account access

    c. Develops password-protected firewall for administrative users to access transaction records and information

3. **Inventory Management Team**
    a. Updates inventory every 2 hours
    b. Have access to inventory and transaction history
    c. Able to read and write data to the database and record/update sales numbers
4. **Finance Transaction Team**
    a. Updates finance data every 24 hours
    b. Have access to transaction history and sales
5. **Test Engineers**
    a. Must report all progress to the project manager by the end of every week
    b. Required to meet with each team at least twice a day and attend all meetings