Clothing Store Point of Sale System
Software Design Specification

By Angela Lee and Anthony Kim
March 7, 2024

System Description:

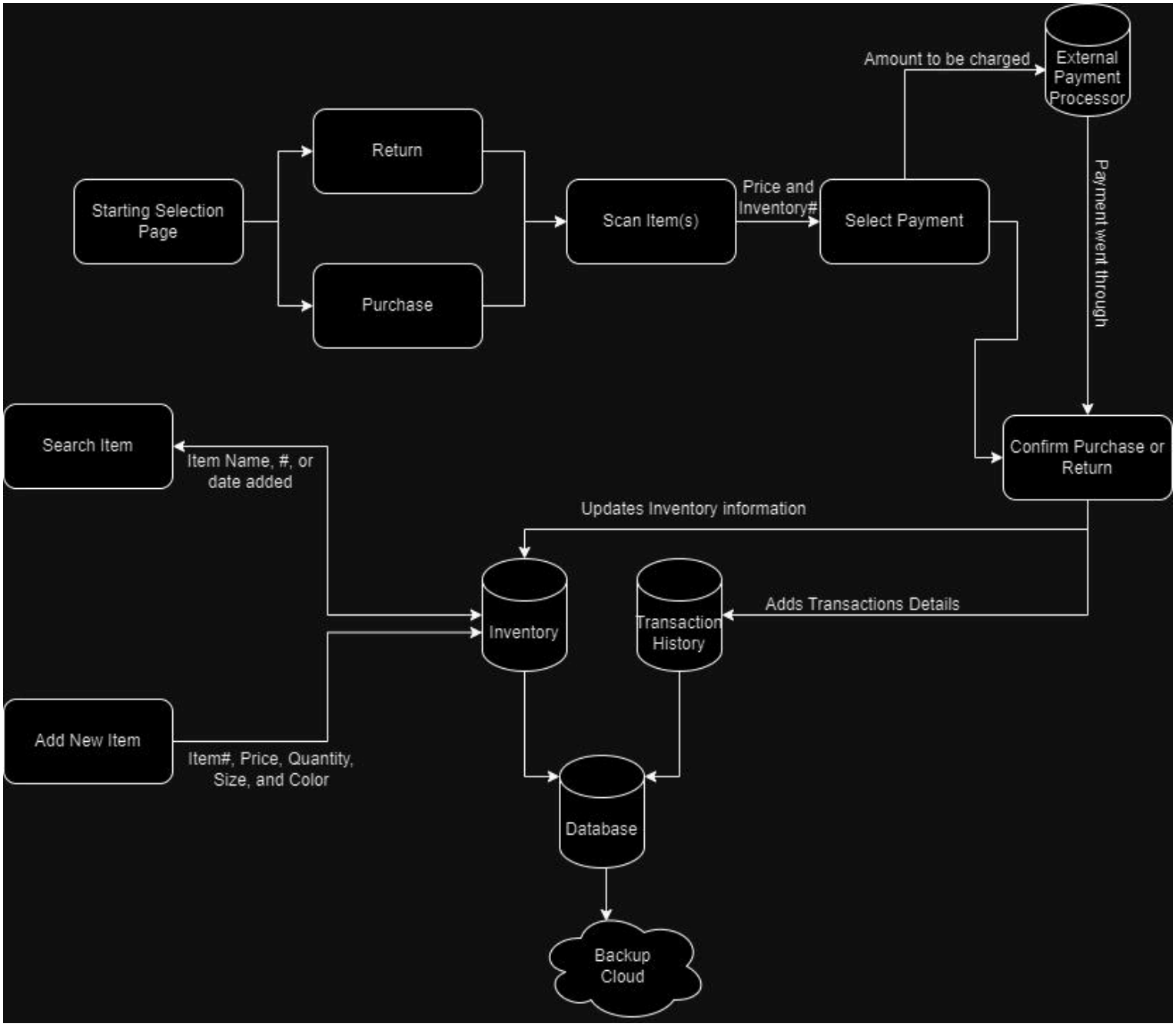
The system software design that we are specifying in this document is a clothing store point of sale system. This point of sale system should serve all of the needs of our clothing store client, as it will be designed to encompass essential functions such as inventory handling and processing transactions and returns.

The system will be integrated throughout all locations of the clothing store, and will be supported electronically on handheld devices by apple and android operating systems. All inventory information will be centrally stored in our client's cloud database, which will be synced across all store locations.

Employees will have access to inventory information and transaction/return processing capabilities, while administrators have additional access to transaction histories and data, thereby maintaining appropriate access control within the system. Customer interactions with the point of sale system will be done through the employees with access to this system, who will be able to make transactions and returns for customers and search for their desired items.

Software Architecture Overview:

SWA Diagram:



SWA Diagram Description:

The Software Architecture Diagram provides a clear view of the system's components and how they interact with one another. The diagram above demonstrates how there are different processes for three main cases. These cases include returns and purchases, searching items, and adding items.

1) Returns and Purchases:

The process for returning and purchasing items can go through a total of 9 components.

1. Starting selection page
2. Selection of either a return or purchase. The choice is then sent to the Scan Item(s) component to signal whether the transaction is a purchase or return.
3. Scan Item(s) will read the item information (price, name, color, and size) from the barcode or the ID# of items. The data is then sent to the Select Payment component.
4. Select Payment will take data given from Scan Item(s) and send only the total price of all the items scanned to the External Payment Processors if credit/debit is selected, it then sends the rest of the scanned item's data to the Confirm Purchase or Return component. If payment was cash or a return was made, it sends Confirmation Purchase or Return the items' data and the transaction information.
5. External Payment Processor takes the data from Select Payment and charges the amount onto the credit/debit card. Once the payment has gone through it sends Confirm Purchase or Return component that the payment has been processed.
6. Confirm Purchase or Return takes the data received from Select Payment and External Payment Processor and confirms that payment has been made, once that has been done it sends all the items' data and transaction details of the purchase or return to two different components. Items' data will be sent to be stored in Inventory and transaction details will be sent to Transaction History.
7. Inventory component will take the item's details (ID#, name, color, size, price) and will add or remove the item from inventory depending on if it was a return or purchase. The Inventory Component will then send a copy of its updated data to the Database component.
8. Transaction History component will take the transaction details given from Confirm Purchase or Return and update itself. After that is done it will send a copy of its updated information to the Database component.
9. Lastly, the Database Component will send its inventory and transaction history data to the Backup Cloud.

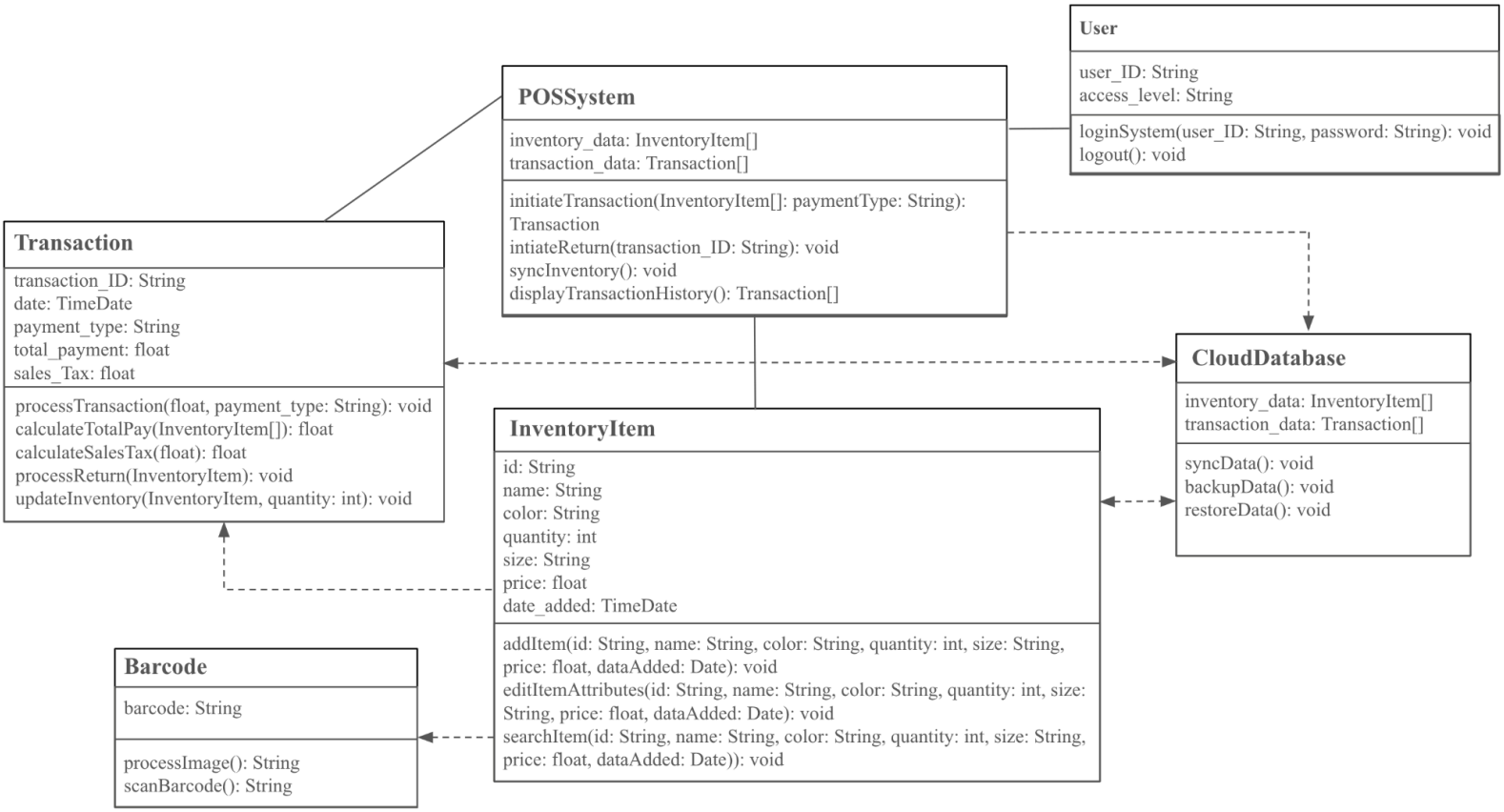
2) Searching Items:

1. Search Item receives an item's name, number, or date added. This data is then sent to the Inventory component.
2. Inventory then takes the data and searches through its data to find if there is a match. Inventory will then send back whether the item they are searching for is available or not to the Search Item component.

Adding Items:

1. Add New Item will receive the necessary data such as item#, price, quantity, size, and color. This data will then be sent to the Inventory component.
2. The Inventory component then takes the data given by Add New Item and updates itself with the new item's data.

UML Class Diagram:



UML Diagram Description:

Description of Classes:

This UML diagram illustrates six key classes of our point of sale system. This includes the Point of Sale System class (POSSystem), the User class, the Transaction class, the CloudDatabase class, the InventoryItem class, and the Barcode class.

1. The POSSystem class encompasses functionalities that involve overall system operations, and also stores inventory and transaction data. This class relies on the Transaction, User, and InventoryItem classes, and it utilizes the CloudDatabase class.
2. The CloudDataBase class is a central hub containing synchronized inventory and transaction data that is securely backed up. It has a reciprocal relationship with the Transaction and InventoryItem classes.
3. The User class defines individuals who are using the system (based on their level of access), along with their login information.
4. The Transaction class outlines how the system processes transactions and handles returns.
5. The InventoryItem class contains and manages inventory information and related functions. It utilizes the barcode class.
6. The Barcode class stores data that is necessary for scanning item barcodes.

Description of Attributes:

- POSSystem class:
    - inventory_data (InventoryItem[]): Inventory_data is an array of InventoryItem objects that represents the current inventory data of the system.
    - transaction_data (Transaction[]): transaction_data is an array of Transaction objects that represents recorded item transactions.
- CloudDatabase class:
    - inventory_data (InventoryItem[]): inventory_data stores inventory data/information into the client's cloud database, which allows for central data management and synching across all store locations.
    - transaction_data (Transaction[]): transaction_data holds a record of every transaction made through the point of sale system.
- User class:
    - user_ID (String): Identifies users of the POS system.
    - access_level (String): Determines level of user's access permissions within the system, depending on whether employee or administrator status.
- Transaction class:
    - transaction_ID (String): Uniquely identifies a transaction.
    - date (TimeDate): The date and time of when the transaction occurred.
    - payment_type (String): The chosen method of payment, such as credit, debit, or cash.

    - total_payment (float): The total amount that will be paid by the customer in the transaction.
    - sales_Tax (float): The sales tax amount to be added to the transaction.
- InventoryItem class:
    - id (String): A unique identifier for each and every item within the inventory.
    - name (String): The name of a clothing item.
    - color (String): The color of a clothing item.
    - quantity (int): The number of items available in stock for a particular item in the inventory.
    - size (String): The size of a clothing item.
    - price (float): The retail price of a clothing item.
    - date_added (TimeDate): The date and time when an item was added to the inventory.
- Barcode class:
    - barcode (String): The unique barcode associated with every item in the inventory.

Description of Operations:

- POSSystem class:
    - initiateTransaction: This begins a new item transaction using a list of items (array) and a specified payment type (String).
    - initiateReturn: Handles the return process for an item(s) from its unique transaction ID (String).
    - syncInventory: Synchronizes the inventory data across the entire point of sale system, which ensures that all locations have up-to-date information on the inventory.
    - displayTransactionHistory: Returns record of item transactions for review, which is only accessible to administrators.
- CloudDatabase class:
    - syncData: This ensures that a certain point of sale system store location's data is synchronized with the client's cloud database.
    - backupData: To maintain safety, this creates backups of the database to prevent loss of data in case of an error.
    - restoreData: Restores the data from data backups, which provides security against possible data corruption/data loss.
- User class:
    - loginSystem: Authenticates a user of the system, which will allow the user access to the system's functions after taking in their user ID (String) and password (String).

    - logout: Ends a user's session in the system, ensuring that access is secured and safely managed.
- Transaction class:
    - processTransaction: This processes the transaction by using details of the sale, such as the transaction amount (float) and payment type (String).
    - calculateTotalPay: Calculates and returns the total amount to be paid (float), including sales tax, based on the items involved in the transaction (array).
    - calculateSalesTax: Calculates and returns the sales tax amount (float) applicable to the subtotal (float) of the transaction.
    - processReturn: Handles the return process of an item by updating the inventory and the transaction history accordingly (InventoryItem).
    - updateInventory: Adjusts the inventory count (int) based on transactions and/or returns (InventoryItem) to ensure that item inventory levels are up-to-date.
- InventoryItem class:
    - addItem: Adds a new item to the inventory with details including the item's ID (String), name (String), color (String), quantity (int), size (String), and price (float). This will update the inventory data immediately.
    - editItemAttributes: Allows item attributes, such as the item ID (String), name (String), color (String), quantity (float), size (String), and price (float), to be modified, which ensures the inventory is accurate and up-to-date.
    - searchItem: Looks for an item within the inventory, which is done using attributes such as item ID (String), name (String), color (String), quantity (int), size (String), and price (float). This is useful for employees when they need to quickly look up and find items for customers.
- Barcode class:
    - processImage: Processes and returns the image of an item's barcode as a String captured by the handheld device's camera.
    - scanBarcode: Interprets the item's barcode from an image to return the associated string of characters.

Development Plan and Timeline:

List of tasks:

- ☐ Designing: Angela
  - ☐ Based on the SWA and UML diagrams, we can start designing the system's software specifications. This includes working on the UI, UX, data structures and algorithms, etc.
- ☐ Development/implementation: Anthony
  - ☐ Writing the system's software code for interface and other components, developing system based on hardware requirements, identifying risks, implementing interaction protocols, working with the cloud system database
- ☐ Testing: Angela
  - ☐ Running through various unit/integration/system test cases and scenarios, verifying and validating elements of the system, identifying and correcting bugs and errors
- ☐ Overview and Deployment: Both
  - ☐ Review of the POS system final product and deployment, implementation of cloud system, software interface, maintenance of security/safety measures and data access restrictions
- ☐ Maintenance: Anthony
  - ☐ Monitor system performance and watch out for issues, carefully access system components, update and improve functionalities, adapt system to new requirements/technological advancements

Timeline:

Start-2 weeks:

The design of software based on SWA and UML diagrams should be completed with a significant amount of detail.

1.5 months later:

Development/implementation will be completed. This will include coding the software's various classes and connecting them to interact with each other properly.

1 month later:

Testing of software will be completed with detailed test cases that test each feature of the system. After test cases are completed any issues with the code will be fixed.

5 days later:

Overview and Deployment will be completed. Will do a thorough overview to ensure the security and reliability of the system. Once the overview is finished deployment of the system will begin.

Deployment-Onward:

Maintenance will begin to check on the stability and security of the system.