Clothing Store Point of Sale System

Team Members: Mark Canilang, Mirek Kupczynski Jr, Angel Guzman

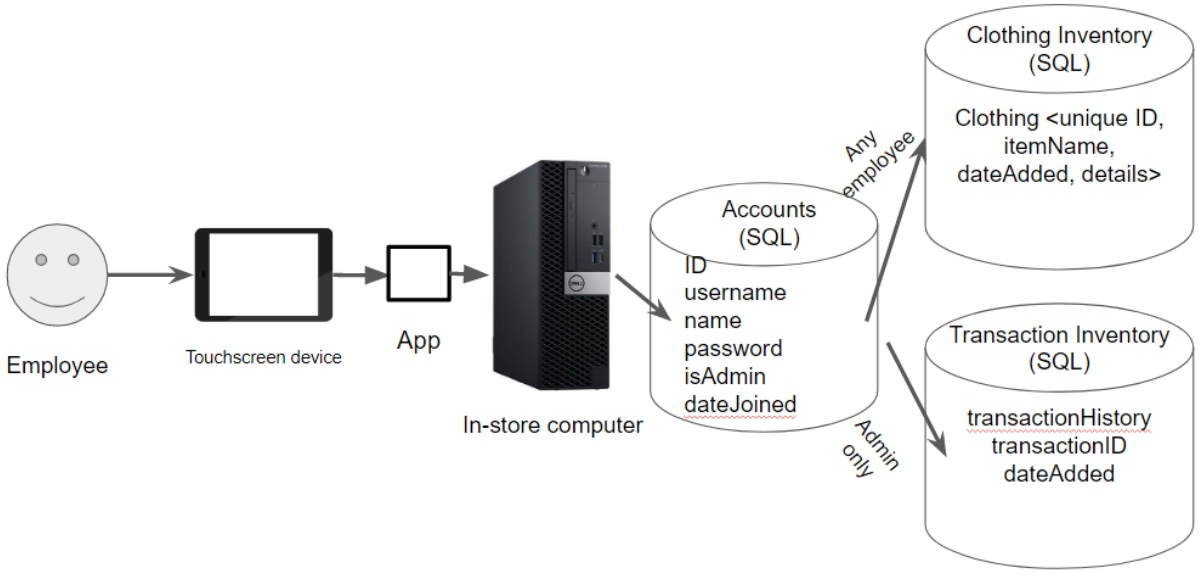
System Description

The system that is being developed here is what is referred to as a point of sale system. This type of system is a computer-based system that helps workers of a clothing store handle transactions and manage inventory, as well as assist management with keeping track of inventory and sales across different stores of the same company.

This system will help streamline many tasks that the workers of stores do frequently day to day. It will make their jobs easier, thus allowing for higher efficiency work to be done, as well as to increase customer satisfaction by giving them a smoother shopping experience.

Software Architecture Overview:

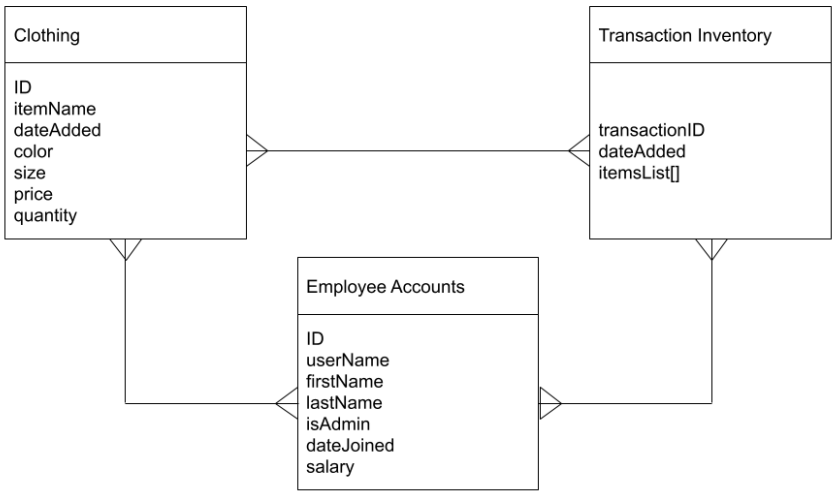
Architectural Diagram Overview:



This diagram illustrates how all major components interact with each other. Employees will interface with the system using the in-store touch screen systems (mainly Apple and Android devices), all of which have an app that acts as the inventory/transaction software. This app will connect to the store's main computer that is accessing the relevant information from the cloud databases. There are three databases, one for the inventory, which all employee users can access, and another will be the transaction database that can only be viewed by administrative users. Finally, there is the account database which stored the credentials for people logging in.

Data Management Strategy

We plan to use three SQL databases: to track clothing items, employee accounts, and transaction history.



Clothing

ID	name	DateAdded	color	size	price	quantity
0	Sweater	2-23-2021	Black	M	$50	20
1	Jacket	6-21-2022	Navy blue	S	$120.99	2
2	Shoes	1-11-2023	Pink	XL	$23.99	43

1) Clothing database

As shown in our SQL table, many fields of clothing are stored in it, such as ID, item name, quantity, etc. This is so that our transactions can access all the clothing information and be able to perform basic actions.

2) Employee accounts

Employee accounts will also be stored in the database. This is to keep track of all the usernames and passwords, so employees can login and manage transactions and check the transaction history.

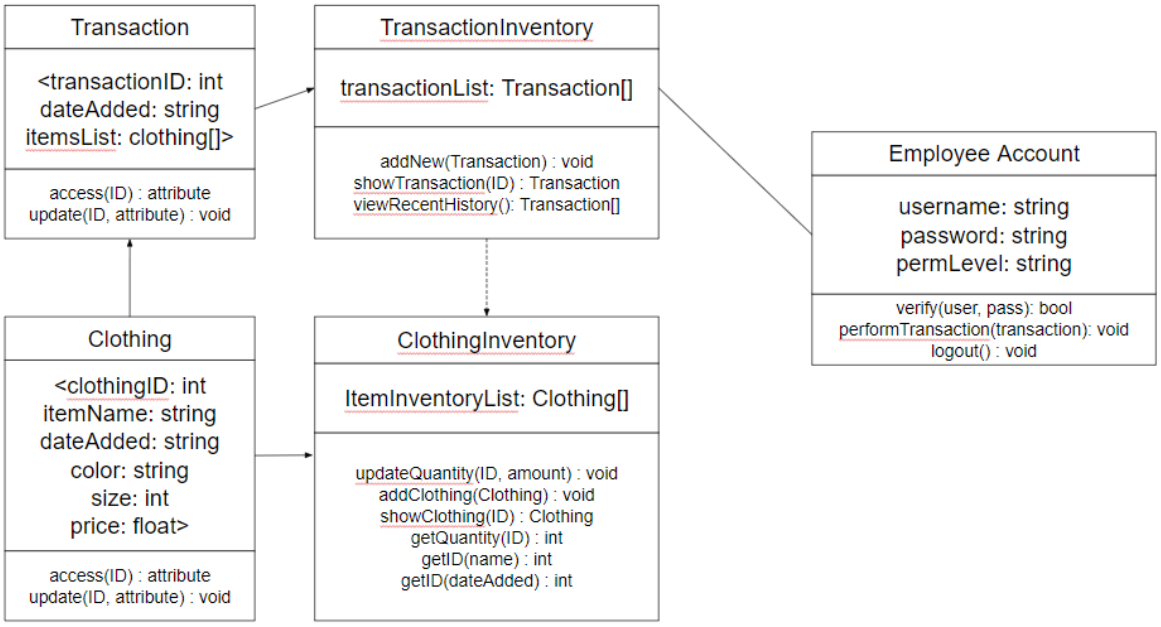
3) Transaction history

The transaction history will store the total price of transactions, the date when they're performed, and a list of the clothing items from a given transaction.

We split it like that accordingly, because those were the three main ideas the client requested us to implement. They all connect to each other, but have individual characteristics that we want to store for each one.

We could have used a noSQL method to store this information, but we chose to use SQL, because the information here is relational with clearly predefined fields we want to use. The data does also not need to be queried super fast, so we thought it'd be best to stick with SQL.

UML Class Diagram:



This diagram illustrates how the primary data structures and classes of this system will operate and interact with each other. There are three primary classes, clothing and transactions are related to the databases, and these will be accessed by the database files, which are accessed by the employee account class.

The clothing classes consists of basic information about individual clothing items, such as:

- unique numerical id (integer)
- the date it was added to the system (string)
- the name of the item (string)
- basic information about the clothing, such as color (string), size (int), price (float),

(Note: different colored versions and different sizes of the same item are considered different, they will share identical information except for the total quantity.)

Next we have the transaction class, which contains its own numerical id, the date of the transaction, and an array of clothing items. The transaction, when being displayed, also shows the total price calculated from its constituent clothing items.

Lastly we have the Employee Account class, which stores expected information such as the username and password (both strings), and an access level, which can range from ordinary employee to administrator. There is also a logout() method to logout once the employee is finished performing transaction actions.

All three of these classes contain the appropriate getters and setters, which have been generalized to simplify the diagram. The databases mentioned above utilize files that contain instances of the two storage classes, with appropriate methods to access and add new items.

We also have two inventories listed here: ClothingInventory and TransactionInventory. TransactionInventory will be the one modified by Employee Account as only logged in employees will be able to both add transactions, view transactions, and view the recent history after logging in and having the correct permissions. Furthermore, transaction inventory will contain the clothing inventory and modify it when corresponding transactions are added or refunded.

To perform these operations:

- addNew(Transaction) will just add a new transaction.
- showTransaction(ID) show prior transactions corresponding to the given ID.
- viewRecentHistory() will display the transactions as a list.

Each of these are associated with the clothing inventory operations:

- updateQuantity(ID, amount) when a transaction is performed.
- addClothing(Clothing) if new clothing needs to be added
- showClothing(ID) to get information about an already added clothing.
- getQuantity(ID) to see the stock.
- getID(name) to view clothing corresponding to an ID.
- getID(dateAdded) to get an ID based on what day it was added.

Development Plan and Timeline:

1- Develop main clothing system (2 weeks) : Mark & Mirek

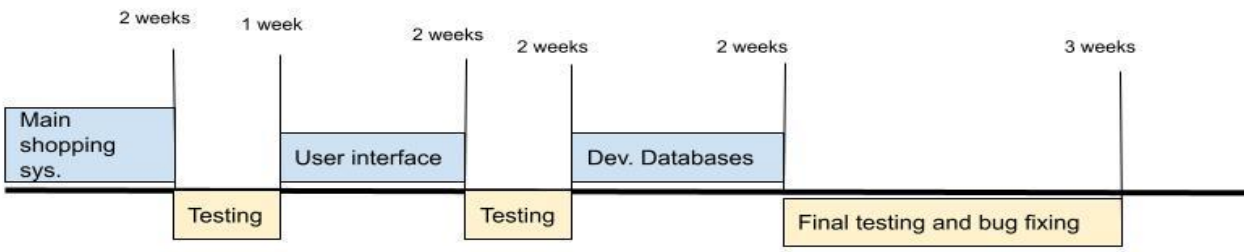
2- Testing main clothing system (1 week) : Mark & Mirek

2- Develop User interface (2 weeks) : Angel & Mark

3- Testing and bug fixing (2 weeks) : Mark & Angel & Mirek

4- Develop databases (2 weeks) : Angel & Mirek

5- Final testing (3 weeks) : Angel & Mark & Mirek



Verification Test Plan:

Test Set 1:

This set of tests involves the clothing related functions and aspects of our software. We'll be testing the functionality of the clothing inventory in tandem with the clothing items, such as getting the clothing ID's and adding new items to the inventory.

Unit Test:
We are testing the functionality of the getID function in the ClothingInventory class.

This function is meant to take a string as an input, and use a search algorithm to match what clothing item in the inventory has the name that was inputted.

We will be using a dummy clothing item for this test that has the name "Shoe" with ID 115.

By entering getID("Shoe"), The program will handle the verification of this input being a string, and the function should return a value of "115".

Functional Test:
We are testing the general functionality of Clothing and ClothingInventory.

This phase of testing will involve adding new clothing items to the inventory. We will add two new dummy items, "Sock" and "Hat" to the inventory, once the inventory already contains 150 items (starting at 0).

Both of the new items will contain all of their pertinent information excluding the ID and dateAdded.

Upon entering addClothing("Sock", (relevant input),), the ClothingInventory will have a new item with an ID of 150, the date added being the current date and time, and a name of "Sock".

Upon entering addClothing("Hat, (relevant input),), ClothingInventory will have a new item with an ID of 151, the date added, and a name of "Hat".

System Test:
For the system test, we will be testing the locally stored ClothingInventory in the store's primary computer and its connection to the tablets in the store that are operated by the employees.

Every time a piece of clothing is searched for by an employee or used in a transaction, the tablet will query the primary computer for the information it needs.

Whenever a transaction alters the quantity of clothing in the inventory, or if the inventory is otherwise changed by an employee during management, the tablet device informs the primary computer of the change, which updates its inventory, and (if necessary) updates information on all tablets currently displaying inventory quantities.

Test Set 2:

This is a test set focused on performing the transaction handling of our software. To be more specific, we'll be testing features, such as creating new transactions, updating transactions, and adding transactions to the TransactionInventory. Furthermore, we will be system testing the Transaction functionality with the EmployeeAccount to ensure that only admin level users can view the transaction history.

Unit Test:

For our second test set, we will first do a unit test of a Transaction. We will make a new Transaction and call its update(). Transactions are randomly assigned a unique ID; for example, it might be 42839515 in this case. A clothing we might update it with is a "black and white dress" labeled "monochrome_dress". We will take this ID, "42839515", and call Transaction.update(42839515, monochrome_dress) and set it to a transaction with a dress.

Functional Test:

Now, we will test the functionality of the TransactionInventory. We will add the previously created Transaction with ID "42839515". We will call TransactionInventory.add() and use Transaction.access(42839515) as a parameter. This should add this previously created Transaction into the TransactionInventory.

System Test:

For our system test, first we will log in using our admin credentials from an Employee account. This will be tested on the employee devices and they will log in. For instance, a set of employee credentials could be "user: yotsubabestgirl pass:quints25252" and they will log in using that. Then, they will perform a transaction with clothing id "monochrome dress" and just input all the transaction info . They might want to verify it was added, so they will click on view history, which calls viewRecentHistory() and displays all that data. Finally, this test will finish with the employee logging out, which calls the logout() function finishing the test.