SDSU | San Diego State University

CS250 Introduction to Software Systems
Professor Miranda Parker

Clothing Store Point of Sales System

[Software Design Specification]

Created by:

Liam Carter

Bryce Jarboe

Andrea Ruvalcaba

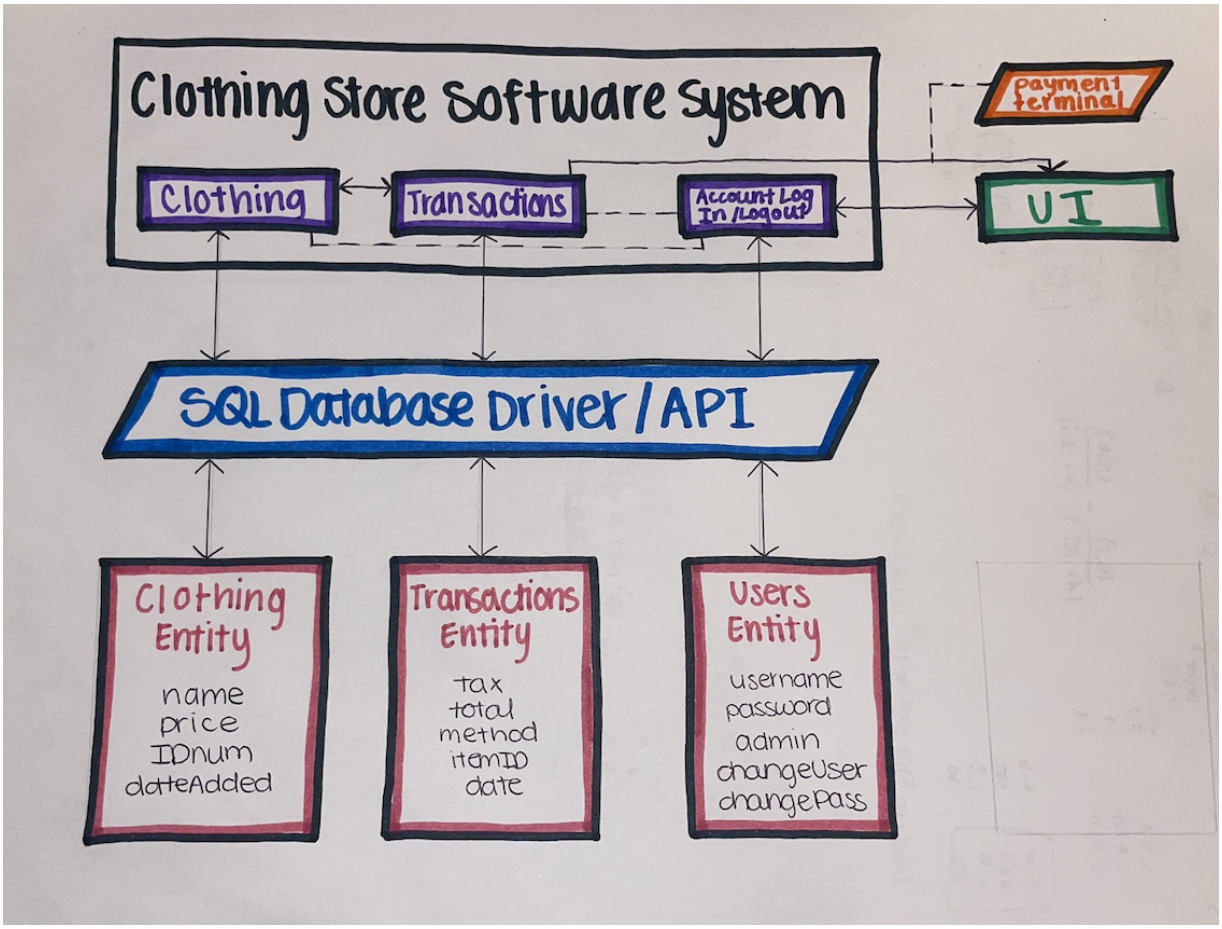
(Group 2)

Table of Contents

Clothing Sale Software System Description

The Clothing Sale Software System will function to streamline clothing store transactions and inventory. The transition towards a software system is necessary in these fields for preventing errors and maximizing efficiency. The system will be able to handle transactions including purchases and returns as well as keeping stock of inventory. Staff members and managers, the primary users of this system, will be able to process transactions, search the current inventory, and add to the inventory, all on their mobile device. Additionally, administrative users will be allowed to access the transaction history made possible with a cloud-backed database. *(from Introduction and Overview, Requirements Specification, by Bryce Jarboe)*

Overview of Software Architecture



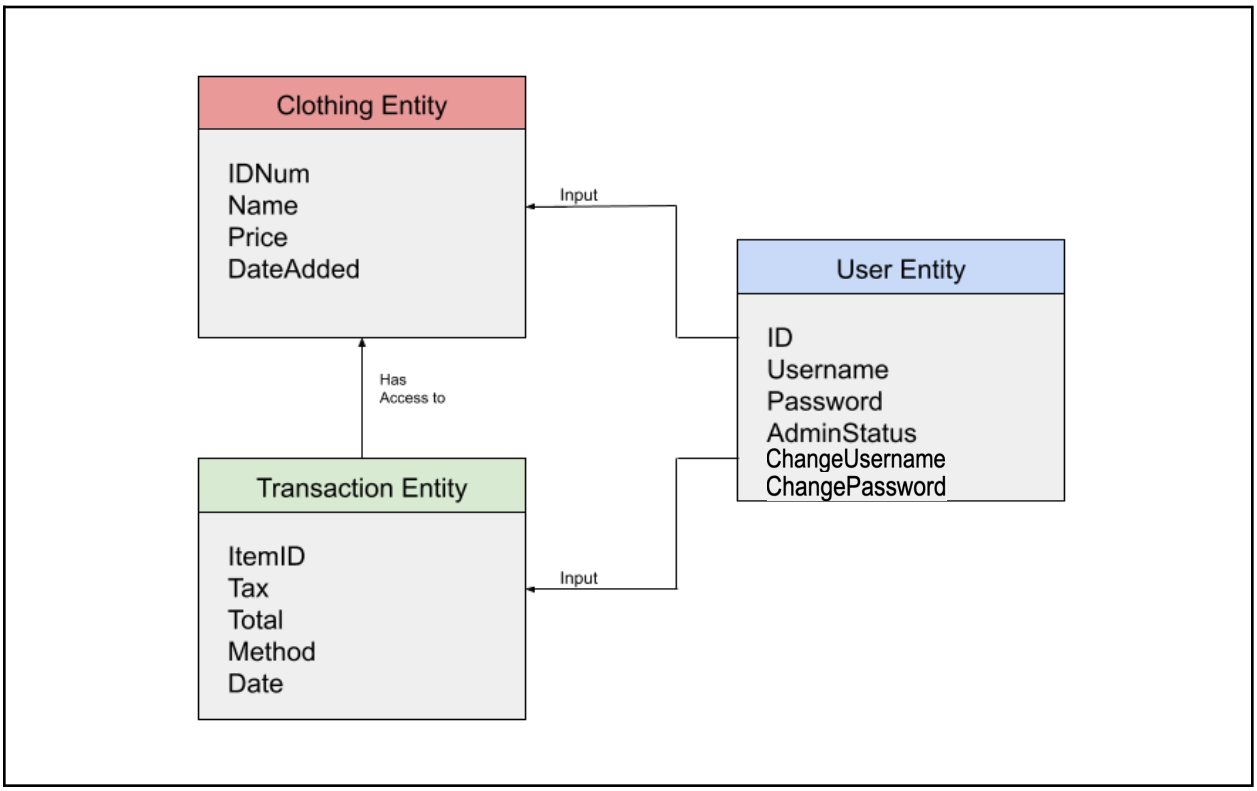
Architectural Diagram

Andrea Ruvalcaba

The following are the core components that service the functionality and operation of the system:

- Clothing/Inventory
    - This component is the central structure that the software system is built on, allowing for the creation and storage of Clothing entries in the respective table/entity via user interaction, and carrying essential data to the transaction processing system.
- Transactions
    - This component takes data from the Inventory through search, and performs transactions via an external terminal, updating the transaction history all while calculating the tax, printing receipts, and mandating the payment method (admin-controlled).
- Account Login/Logout
    - This component provides users with sophisticated interaction as well as added security to the system, maintaining an entity of user accounts that hold usernames and passwords, administrative status, automatic logout of idle users, option to change a username or password, and payment management.
- SQL Database
    - The data in question will be stored under a SQL database management system. See *Data Management* section for details.

Data Management



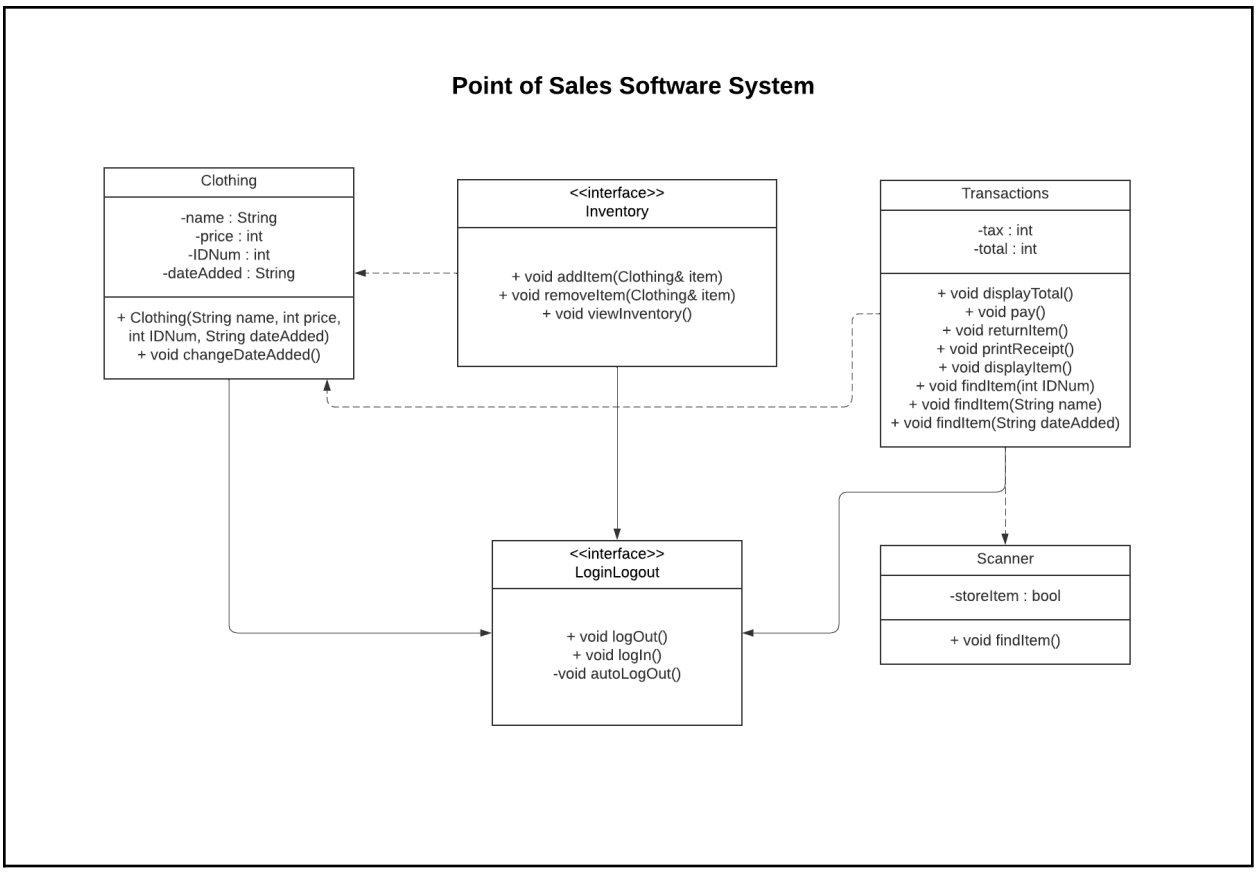
Entity Relation Diagram
Bryce Jarboe

The software system will manage data through the traditional SQL model via the integration of a SQL database driver or API, the choice of which depends on the development process and test results. Methods within the system's main classes that modify necessary data will be capable of issuing SQL statements through the driver/API and thus reading and writing to the database directly.

As for the organization of the data itself, the database will be divided into 3 main entities (tables) and exhibit the following behaviors:

- Clothing:
  - Indexed by ID Number
  - Fields: Name, Price, DateAdded
- Transactions:
  - Indexed by ID Number
  - Fields: Total, Tax, Method, Date
- Users:
  - Indexed by arbitrary ID
  - Fields: Username, Password, AdminStatus, ChangeUsername, ChangePassword

UML Class Diagram



UML Class Diagram

Liam Carter

Clothing

This class will create the bulk of the primary data. Each Clothing object will hold data that, in tandem with the other classes, will prove essential to the cohesion and overall functionality of the system. See below for details.

- name : String
  - The *name* attribute is a means of identification for the inventory's search functionality.
- price : int
  - The *price* attribute (in cents) will be carried through transactions to the payment terminals, being what is effectively charged and/or refunded.
- IDNum : int
  - The *IDNum* attribute is another means of identification for the inventory's search functionality. In addition to inflating the identifiable data, the int data type would allow for hashing (i.e. faster search times).
- dateAdded : String
  - The *dateAdded* attribute, while serving as an additional means of identification, holds data of interest to the inventory and transaction history, keeping records of each entry regarding when it was added.
+ Clothing(String name, int price, String dateAdded)
  + The instantiation of a new Clothing object will read user input and implement the respective parameters. Its constructor will generate an IDNum not currently used throughout the current inventory.
+ void changeDateAdded()
  + *Administrative Access Only*
  + This setter allows Admins to modify the *dateAdded* attribute manually, should it be necessary.

Inventory

Inventory is an interface connected to the Clothing class that can access Clothing objects in order to modify the current inventory of the store.

+ void addItem(Clothing& item)
  + *Administrative Access Only*
  + This will allow admins to add to the current inventory of items, mainly to be used when new products arrive

+ void removeItem(Clothing& item)
    + *Administrative Access Only*
    + This will allow admins to remove from the current inventory of items, to be used when there are defective products that need to be removed.
+ void viewInventory()
    + *Administrative Access Only*
    + This lets admins view the current inventory of a store but does not allow changes to be made from there.

LoginLogout

LoginLogout is an interface that is used by all other classes because it needs to be able to be accessed at any time.

+ void logOut()
    + Allows the user to log out of the system at any time, even in the middle of a transaction.
+ void logIn()
    + Allows the user to log into the system with a unique password tied to their user ID.
- void autoLogOut()
    - This function logs the user out if they do not interact with the system for a set period of time in order to maintain security.

Scanner

Scanner is the primary tool that the system uses in order to identify any clothing items that are presented. It is used to scan barcodes and instantly know all the attributes of the clothing item

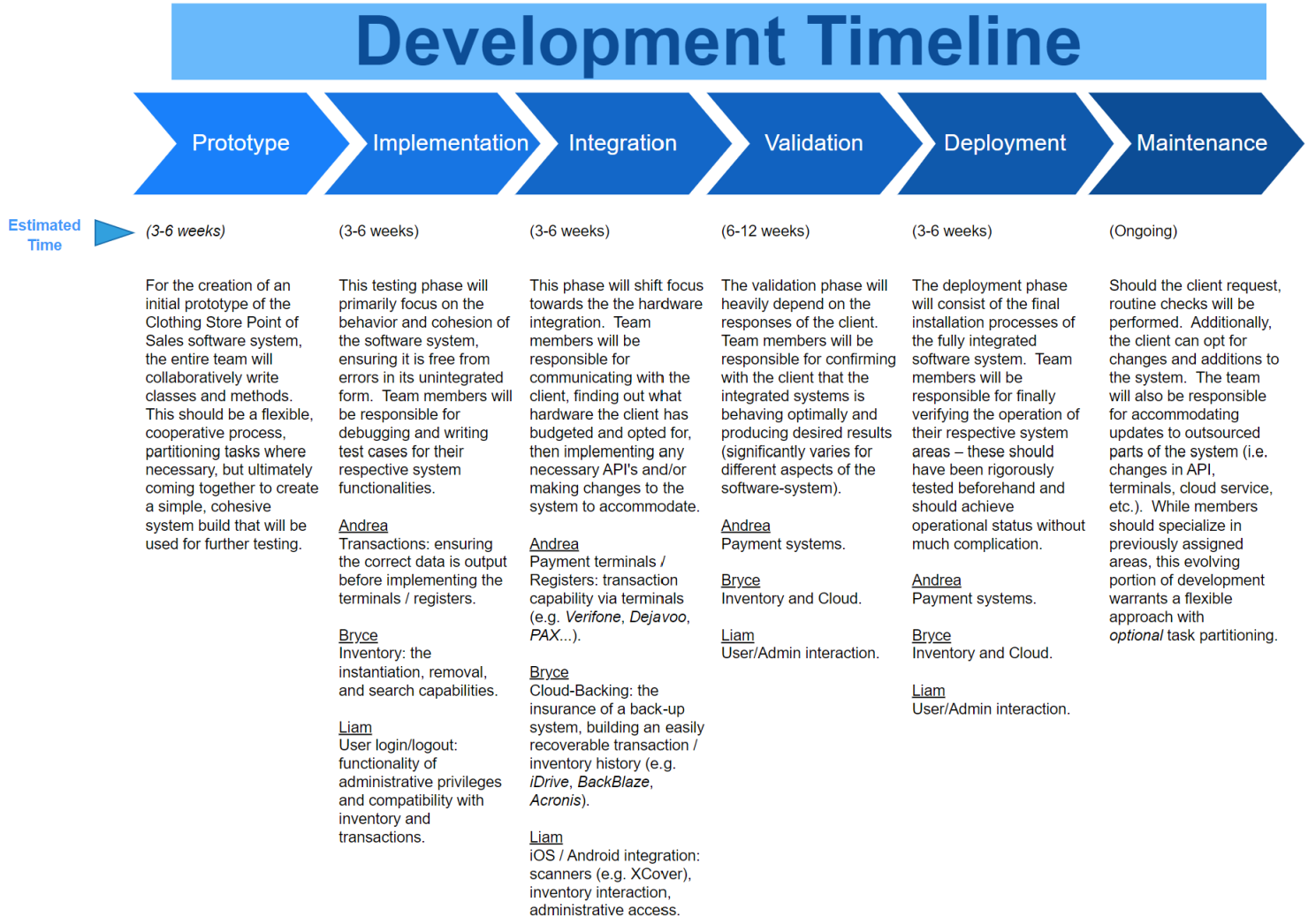
- storeItem : bool
    - The storeItem variable shows if the barcode scanned is a item found in the store or not. This is used in order to prevent confusion.
- void findItem()
    - This method is called every time a barcode is scanned. It searches a map of all the items in the store using the *IDNum* and if the item is not found it will set the *storeItem* to false and will let the Transactions class know there was an issue finding the item.

Transactions

- tax : int
    - The *tax* variable is programmed to have the correct state tax based on the location of the store. The tax is used in the final calculation of the total.
- total : int
    - The *total* variable is used to calculate the total cost of all the items in a given transaction, add them all up, making sure to apply tax and any other partial payments like coupons or gift cards.
+ void displayTotal()
    + displayTotal() is automatically called after each item is scanned to show the current subtotal for the current transaction.
+ void pay()
    + Adds the option for a worker to select that lets the customer pay. It would be bad otherwise if the customer could pay at any time during the transaction as it would result in much confusion.
+ void returnItem()
    + Similar to the pay() function. It can be selected after all items have been scanned as well as a receipt and will refund the items to the card that bought it. Or cash could be refunded as well if the first does not apply.
+ void printReceipt()
    + Automatically called at the end of a transaction and prints a receipt for the customer to keep.

+ void displayItem()
    + Like displayTotal(), this method is called after every item is scanned, but instead it shows all relevant information about the item scanned to the customer such as price and name.
+ void findItem(int IDNum)
+ void findItem(String name)
+ void findItem(int dateAdded)
    + The findItem functions are all the same function that can be called with different parameters through the use of overloading. These functions are used to find the price and other info of an item whose barcode will not scan or is missing. The clothing can be found via *IDNum*, *name*, or *dateAdded*.

The following timeline provides information regarding the specific phases of development, an estimation of how much time they will take, and the partitioning of tasks among team members.



Development Timeline
By Bryce Jarboe

Authorship

Andrea Ruvalcaba

System Architecture Diagram

System Architecture Descriptions

Bryce Jarboe

System Description

Data Management

Development timeline

Liam Carter

UML Diagram

Class Descriptions