COMP 370 Group Project

Group 10

Navjot Bhatti, Hardeep Sandhu, Jonathan Fitz

December 9, 2018

To Dr. Ali,

Our submitted project was not at the level of competency that we expect of ourselves.

Since you allowed us to resubmit the project again through blackboard (this time including the source code!), I took this as an opportunity to tweak the previously submitted Final Report. My hope is that *everything submitted on time be treated as such*, and that only the modifications made after the deadline (detailed below) be penalized with late marks.

If you agree, I have carefully listed the modifications made to the original submitted Final Report to create the document you are now reading:

Modifications to Final Report **After** Deadline:

- Table of Contents and page numbers were added
- Database Schema diagram was tweaked to improve style consistency
- Less detailed class diagram was created to increase readability
- Original (detailed) class diagram was reformatted and included in .zip file
- Final Thoughts section was added
- Signed statement of originality was added
- Notable Tools section was added
- References section and two APA references were added

Even though there were other changes I was tempted to make (ex. reorganizing the test cases), I wanted to make it as easy as possible for you to understand what exactly I changed.

Therefore, the changes described above are the **only** modifications that have been made to the original Final Report, and you can verify this by downloading the originally submitted report from our GitHub repository: https://github.com/boltu11/DashForCash.git

I hope you enjoy reading the rest of this report.

Table of Contents

Individual Contributions Breakdown

	Navjot Bhatti	Hardeep Sandhu	Jonathan Fitz
System Requirements	17%	17%	66%
Architectural Diagrams	0%	0%	100%
Use Case Diagram + Use Cases	10%	0%	90%
Sequence Diagrams	33%	66%	0%
Class Diagrams and Interface Specification	50%	0%	50%
User Interface Design and Implementation (Frontend)	50%	50%	0%
Backend Programming	100%	0%	0%
Design of Tests + Executing Tests	0%	0%	100%
Creating PowerPoint	0%	0%	100%
Writing Report (ie. Filling in the gaps)	15%	0%	85%

Management Style

Our group attempted to combine our favorite aspects of the Waterfall method with those from Agile. For example, we generally followed the linear progression recommended by Waterfall (Requirements first, then Design, Development, and Testing). However, our small size allowed us to document a lot of our work verbally. We also let people have quite a bit of freedom in how they accomplished their tasks, and even what tasks they would undertake.

Customer Requirements

Point of Sale (POS) describes the moment when a customer is given goods in exchange for payment. A POS system handles this transaction.

It is not practical for most businesses to create their own POS system. Thus, these businesses must purchase this system from an outside source.

For our project, we implemented a general POS system designed to be used by a business such as this. Our system is *general*, in that it allows the business to choose the items that can be purchased from this POS system. This is practical for the business that deploys this system; the same POS system can be reconfigured to work in multiple scenarios. For this same reason, creating a POS system that can be handled by a wide range of businesses is cost-efficient.

A database was used hold information needed by all users of this system. Thus, the database holds three key pieces of information:

- List of goods sold by the store and their associated price
- The names of employees at the store, as well as their login password, login ID, and employment rank (cashier or manager)
- Previously saved transactions that can be used to restore the progress of the transaction.

We assume that the business does not require barcodes. Examples of this kind of business include coffee shops and movie theaters. The cashier adds the customer's desired good by selecting on the category icon of that item.

Glossary of Terms

System - Refers to the implementation of this POS software

Client - Represents someone using the system. Can also refer to the computer that the system is installed on.

Database - Entity that stores information available to all clients. Usually this would be run on a computer physically separate from a client. The database is usually considered to be part of the system, but occasionally it was found convenient to separate the two.

Customer- A person who wishes to purchase goods from a seller.

Virtual Cart (or simply Cart) - List of goods entered into the system representing the items that a single customer wishes to purchase.

Transaction - General term referring to all actions involved in the exchange of goods between a customer and a seller.

Cashier - Person who uses the system to manage the transaction between the customer and the seller.

Manager - Person who has the authorization to add and remove cashiers to the system.

System Requirements

The word “shall” is used to describe requirements that the system *must* implement, whereas the word “should” is used for requirements that the system is simply *recommended* to implement.

Enumerated Functional Requirements

Requirement ID	**Description of Requirement**
REQ_F1	The system shall ensure that a valid user ID and password be provided to the system before the user can access the rest of the system.
REQ_F2	The system should prevent anyone from logging in to the system for 10 minutes if more than 5 incorrect login attempts are made within a span of 10 minutes.
REQ_F3	The system shall uniquely identify a user by their user ID. A user with a user ID is said to be ‘registered.’
REQ_F4	The system shall allow a registered manager to add and remove a cashier in the database
REQ_F5	The system shall provide a unique Login ID to every manager or cashier added to the database.
REQ_F6	The system shall allow a registered user to change their password.
REQ_F7	The system shall allow a registered manager to add or remove from the database the type of goods sold by store.
REQ_F8	The system should allow the manager to group all items sold into categories, and these categories into subcategories, and so on.

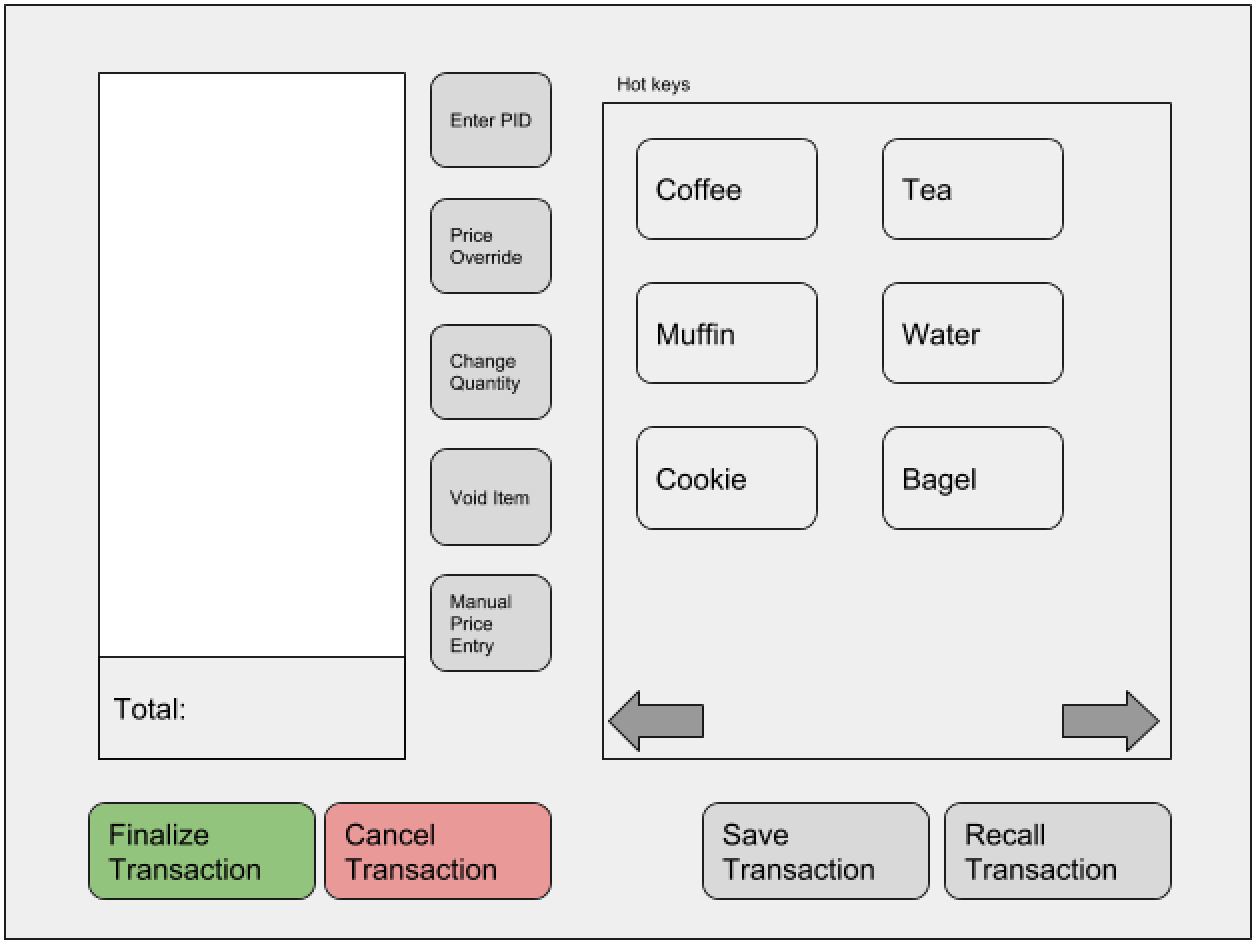
REQ_F9	The system shall allow the cashier to add items to a customer’s virtual cart.
REQ_F10	The system shall allow the cashier to add an item to the customer’s virtual cart by selecting the category that the item belongs to.
REQ_F11	The system shall allow the cashier to change the quantity of an item in the virtual cart.
REQ_F12	The system shall allow the cashier to override the price of an item in the virtual cart. This is useful when an item is priced incorrectly.
REQ_F13	The system shall allow the cashier to add an item to the virtual cart by manually typing in the product id (PID).
REQ_F14	The system shall allow the cashier to remove an item from the customer’s virtual cart.
REQ_F15	The system should allow the cashier to charge the customer without adding an associated item to the customer’s virtual cart. This is useful when the cashier is not sure which category an item belongs to but remembers the price of the item.
REQ_F16	The system shall allow the cashier to be able to cancel the transaction at any point until the transaction has completed.
REQ_F17	The system shall allow the cashier to save the current transaction to the database so it can be resumed in the future.
REQ_F18	The system shall allow the cashier to retrieve a past transaction from the database if it was saved, so that the transaction can be resumed.
REQ_F19	The system should allow the cashier to check the price of an item. This is convenient, but not something considered necessary.
REQ_F20	The system shall ensure that the current total price always equal the sum of the prices of the items in the customer’s virtual cart.
REQ_F21	The system shall ensure that the virtual cart only ever contains items associated with one customer.
REQ_F22	The system shall allow the cashier to finalize the transaction and take payment from the customer.
REQ_F23	The system should allow the cashier to print a receipt of the purchase. This may not be necessary when the seller only accepts cash.

Enumerated Non-Functional Requirements

Requirement ID	**Description of Requirement**
REQ_NF1	The system shall ensure that only authorized users can access the system.
REQ_NF2	A user should be logged out of the system after 5 minutes of idle time.
REQ_NF3	The system shall be created using the Java programming language.
REQ_NF4	The database shall be created using MySQL.
REQ_NF5	The system should display requested information to the user within 0.5 seconds of the cashier’s request.
REQ_NF6	A cashier with 0.5 hours of training should be able to add a random item to a customer’s virtual cart within 8 seconds, on average.

On-Screen Appearance Requirements

The following is an early mockup of the screen that a cashier would use to input customer orders into the system (ie. to create the customer's virtual cart).



Stakeholders

- Customer
- Cashier
- Manager

Actors and Goals

Actor	Actor's Goals
Cashier	Oversee transaction between customer and seller

Manager	Adds and removes cashiers from the database, adds to the database the goods sold by the store
Customer	Receives goods after purchase when transaction is complete

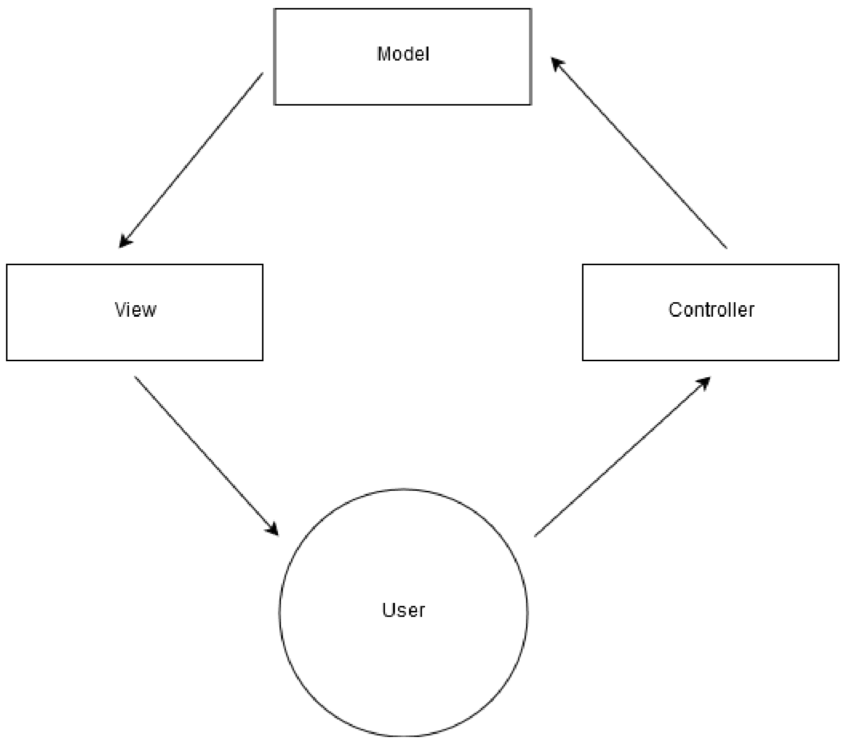
Architecture Design

The purpose of architectural design is to bridge the gap between the requirements process above and the more detailed system design below.

Model-View-Controller Pattern

The Model-View-Controller (MVC) Pattern was chosen as the basis of our architecture for a few reasons: In this model, the Controller, Model, and View objects do not know or care about the inner workings of the others; all they need to do is satisfy the interface of the object that they want to communicate with. This tends to lead to more modular code, which means that code can be more easily replaced.

Another advantage of this pattern is that it is popular, which means it is easy to find examples of code done in this style. This is a legitimate advantage and a valid reason for choosing an architectural pattern.

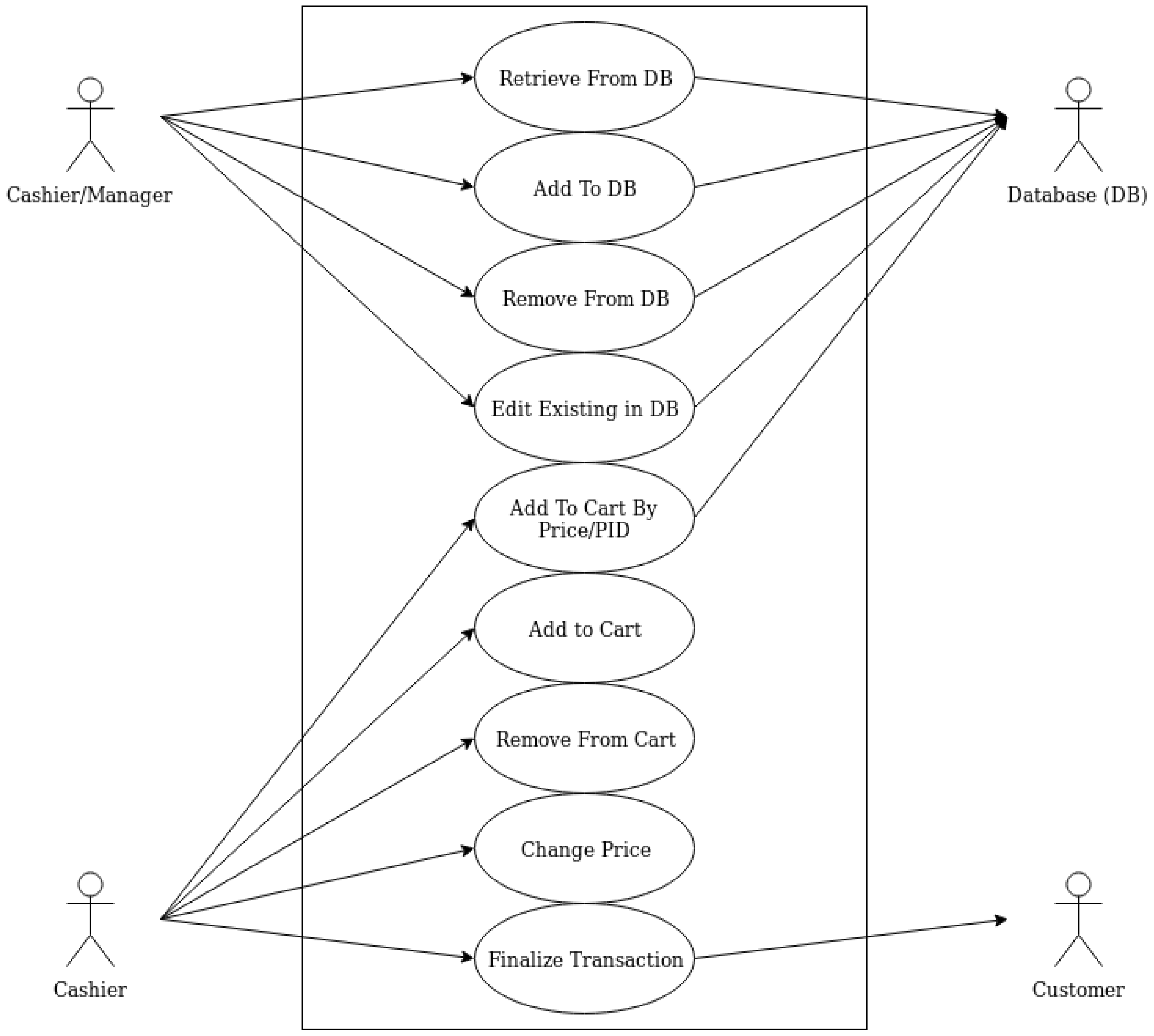


Interaction Diagrams

While use cases can be used to elicit requirements, we followed the approach suggested by Sommerville, and used use cases in conjunction with sequence diagrams to create interaction diagrams for our system (Sommerville, 2016).

Use Cases

Graphical Description



The above diagram shows the common workflows between the system and the external user. The rectangle in the middle represents the system. Notice here that the Database is considered not to be part of the system; this was done for clarity.

Detailed Descriptions

The use cases described in the use case diagram above are explained here in more detail.

Use Case 1: Retrieve From DB
Actors: Cashier/Manager, Database
Description: A user may retrieve information from the database.
Data: Goods sold by the store, saved transactions, employee information
Stimulus: Request entered by user
Response: The requested information

Use Case 2: Add To DB
Actors: Cashier/Manager, Database
Description: A user may add information to the database.
Data: Goods sold by the store, a new transaction, employee information
Stimulus: Request entered by user
Response: Confirmation that the database has been updated

Use Case 3: Remove From DB
Actors: Cashier/Manager, Database
Description: A user may remove information from the database.
Data: Goods sold by the store, saved transactions, employee information
Stimulus: Request entered by user
Response: Confirmation that the database has been updated

Use Case 4: Edit Existing In DB
Actors: Cashier/Manager, Database
Description: A user may edit existing information in the database.
Data: Goods sold by the store, employee information
Stimulus: Request entered by user
Response: Confirmation that the database has been updated

Use Case 5: Add To Cart By Price/PID
Actors: Cashier, Database
Description: A cashier may add an item to a customer’s virtual cart by either typing in the price of the item or entering in the product ID (PID).
Data: The item associated with the PID
Stimulus: Request entered by cashier
Response: Item is added to customer’s virtual cart. Sum of prices of items in the cart is updated.

Use Case 6: Add To Cart
Actors: Cashier, Database
Description: A cashier may add an item to a customer’s virtual cart by recursively selecting the item category icon until the exact item icon is shown.
Data:
Stimulus: Request entered by cashier
Response: Item is added to customer’s virtual cart. Sum of prices of items in the cart is updated.

Use Case 7: Remove From Cart
Actors: Cashier
Description: A cashier may remove an item from a customer’s virtual cart.
Data:
Stimulus: Request entered by cashier
Response: Item is removed from customer’s virtual cart. Sum of prices of items in the cart is updated.

Use Case 8: Change Price
Actors: Cashier
Description: A cashier may change the price of an item in a customer’s virtual cart.
Data:
Stimulus: Request entered by cashier
Response: Price of selected item in virtual cart is changed. Sum of prices of items in the cart is updated.

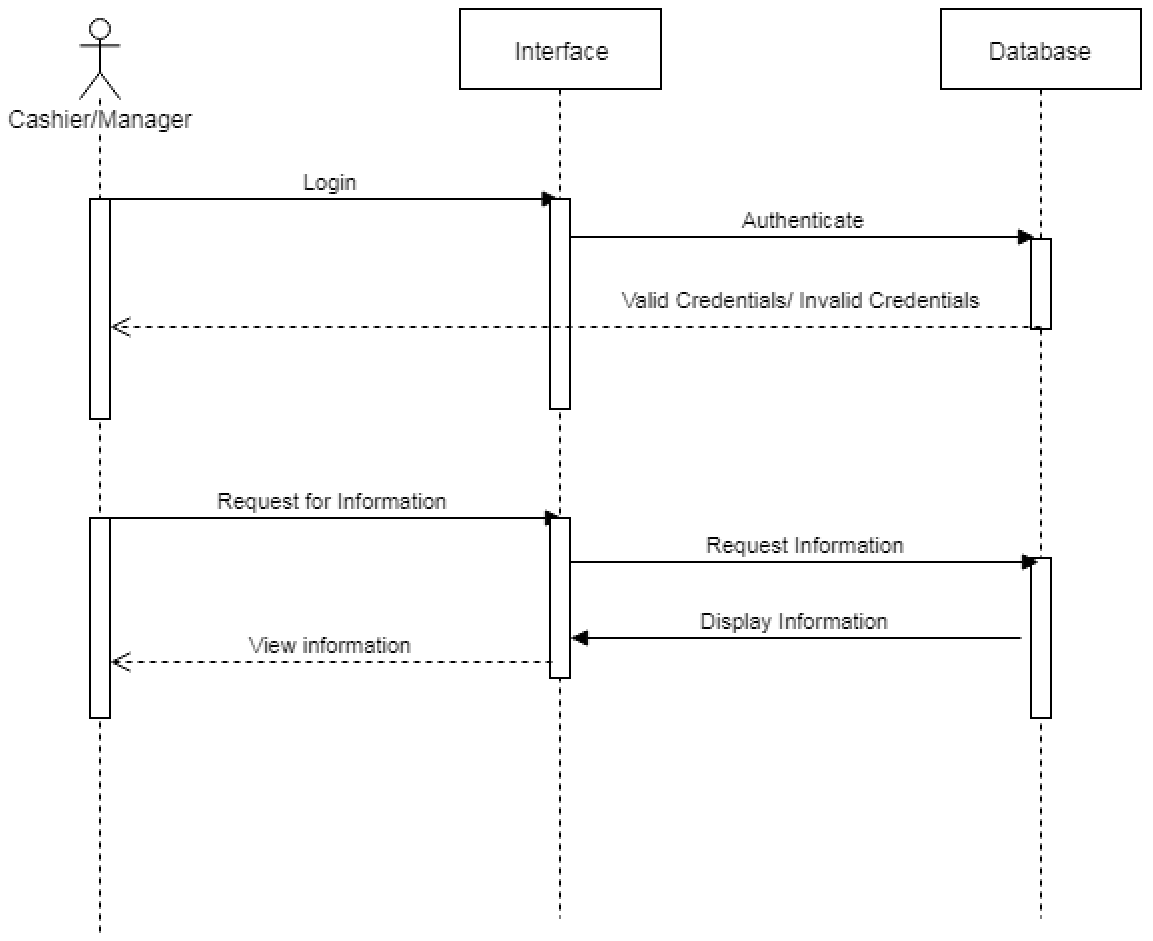
Use Case 9: Finalize Transaction
Actors: Cashier, Customer
Description: A cashier may accept payment from the customer to end the transaction
Data:
Stimulus: Request entered by cashier
Response: Payment from the customer is received, and the transaction is complete.

Sequence Diagrams

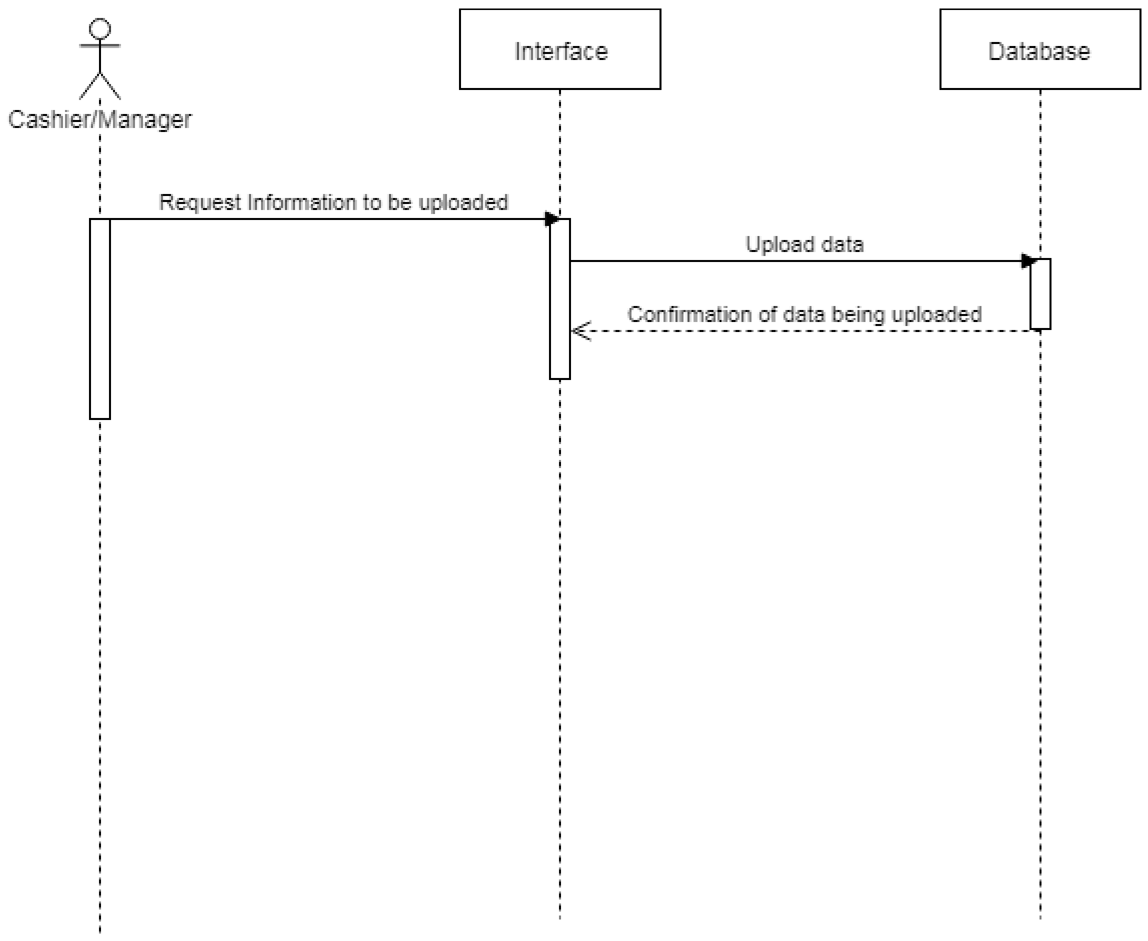
Use Case System Sequence Diagrams

Each of the use cases are now examined from a different perspective. Instead of focusing on interactions between the system and external users, the interactions between system components are modelled.

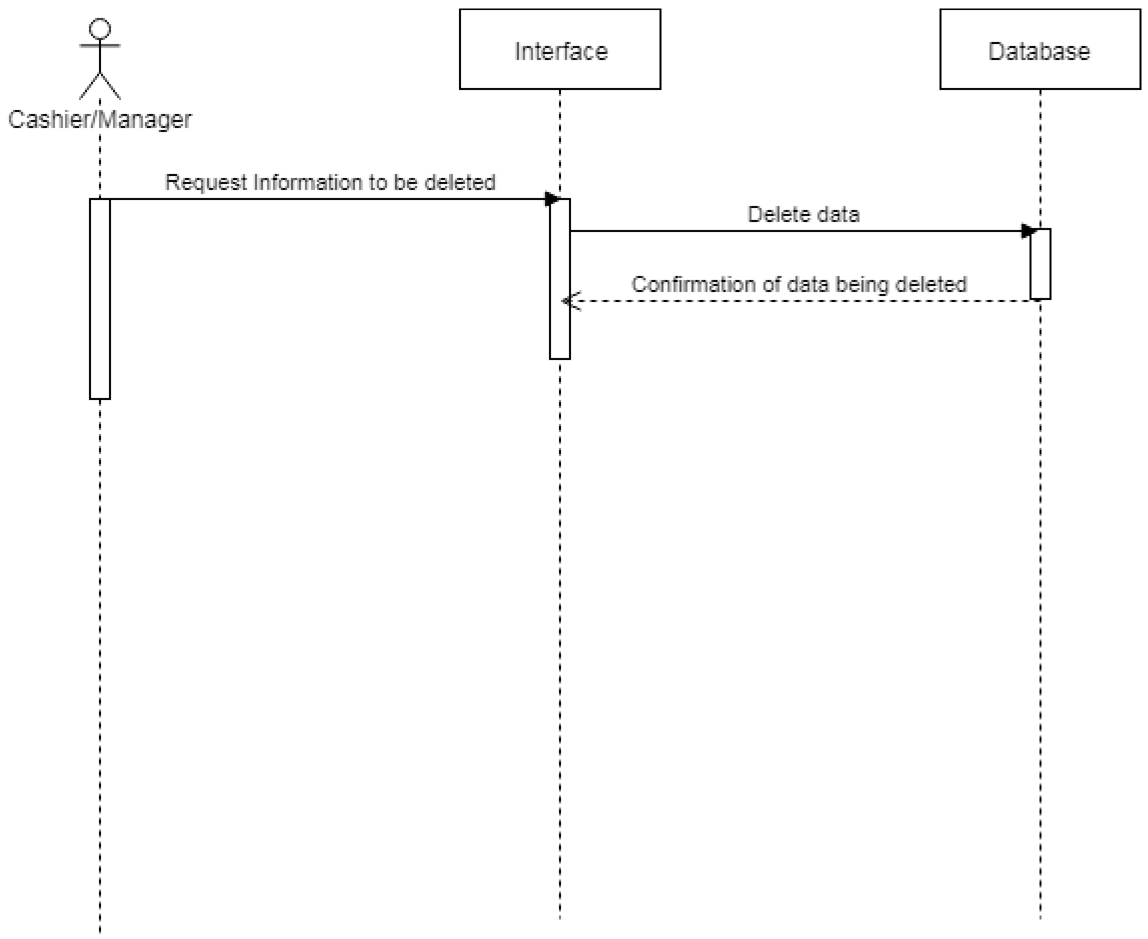
Use Case 1 - Retrieve From DB



Use Case 2 - Add To DB

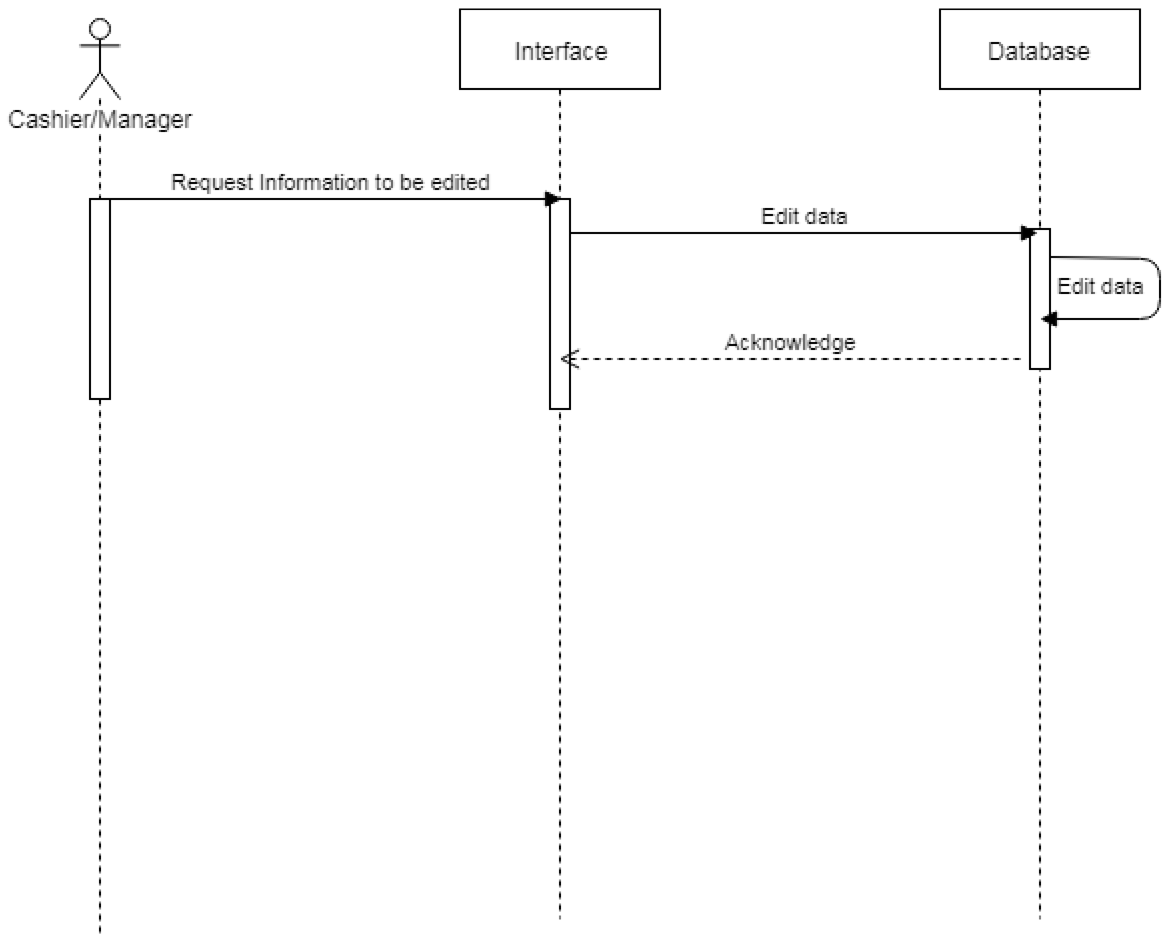

Cashier/Manager
Interface
Database
Request Information to be uploaded
Upload data
Confirmation of data being uploaded

Use Case 3 - Remove From DB

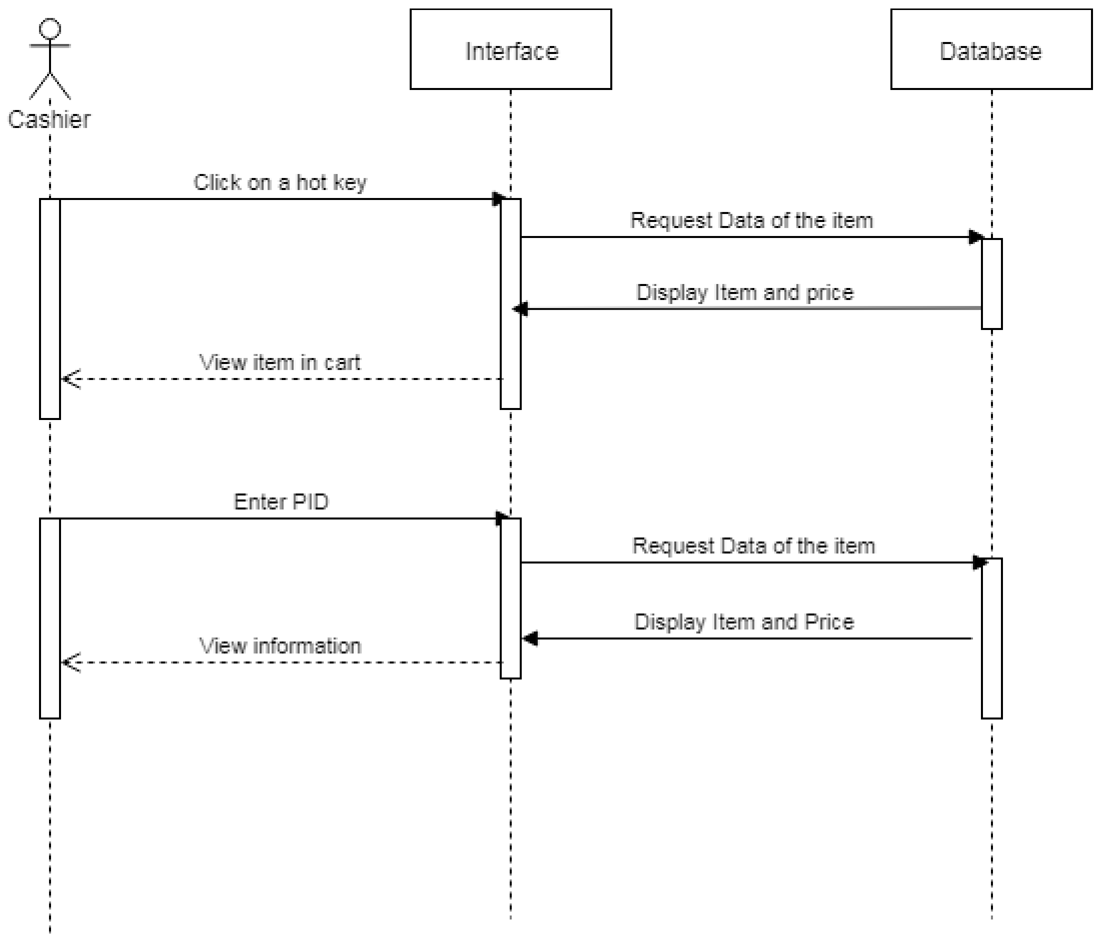


Interface
Database
Cashier/Manager
Request Information to be deleted
Delete data
Confirmation of data being deleted

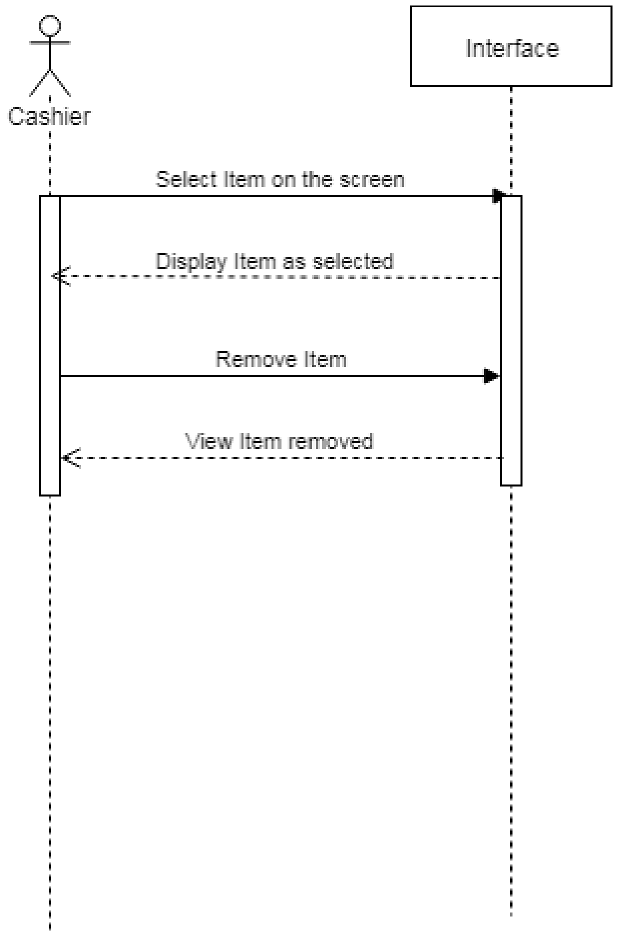
Use Case 4 - Edit Existing in DB



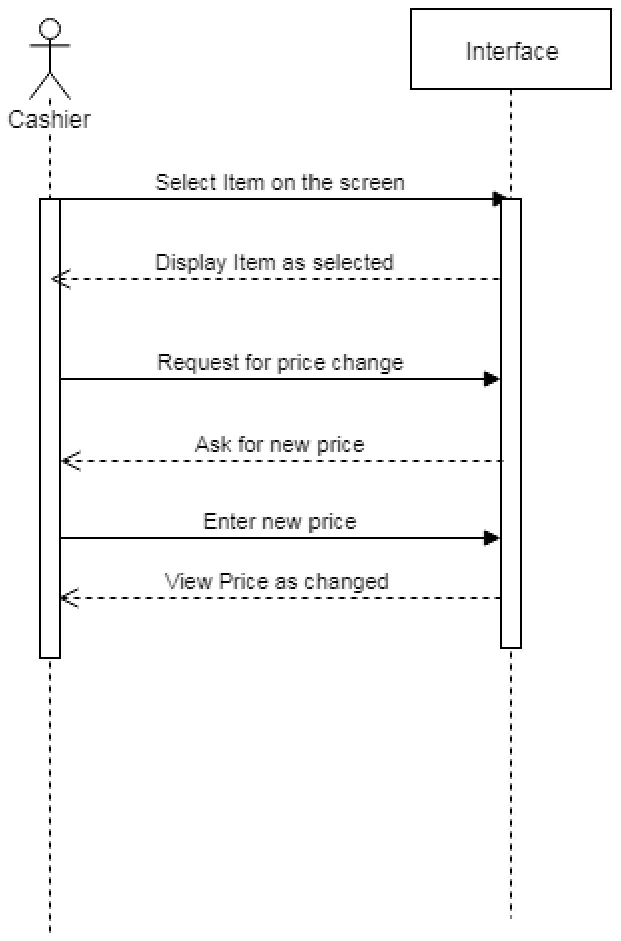
Use Case 5 - Add to Cart By PID/Price



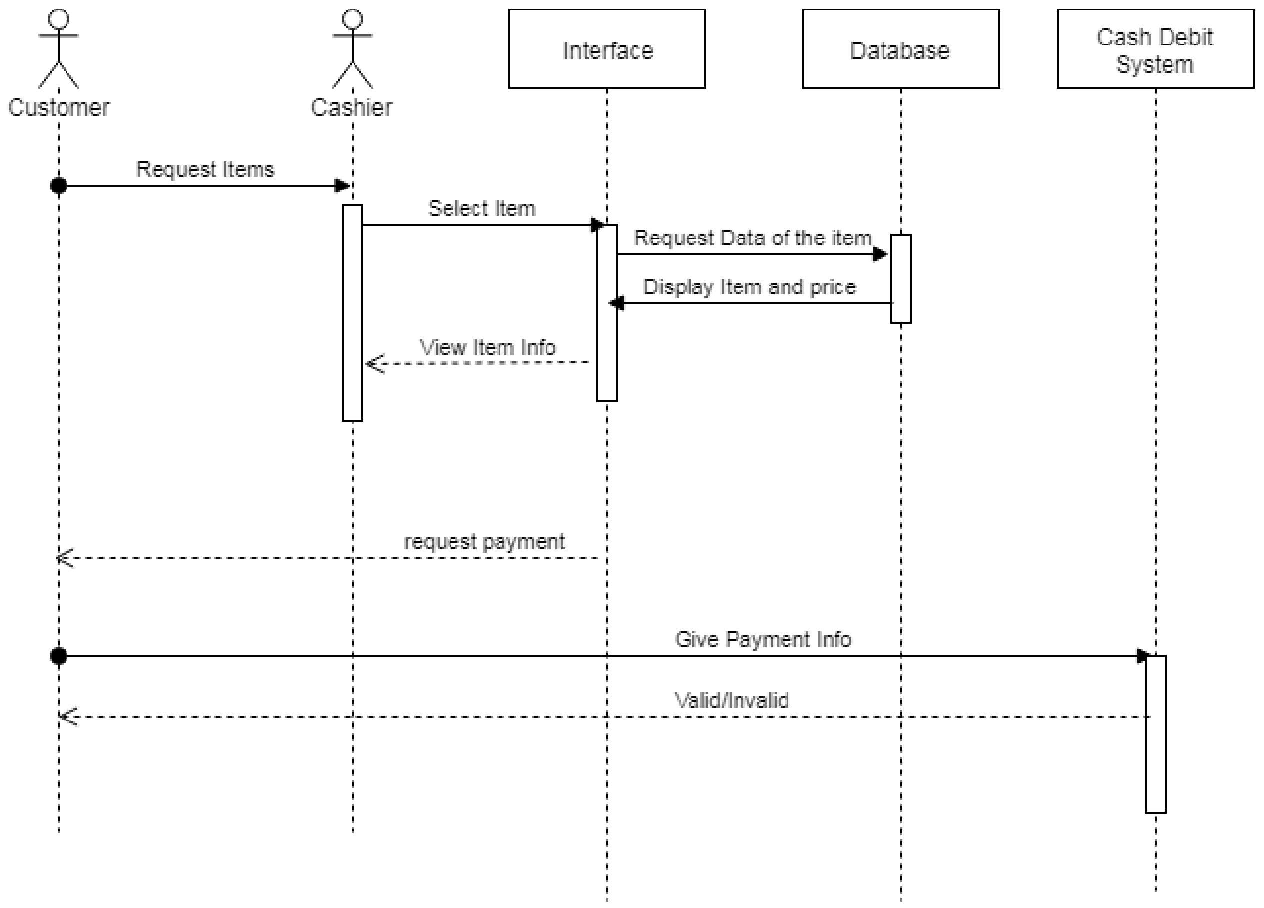
Use Case 7 - Remove From Cart



Use Case 8 - Change Price



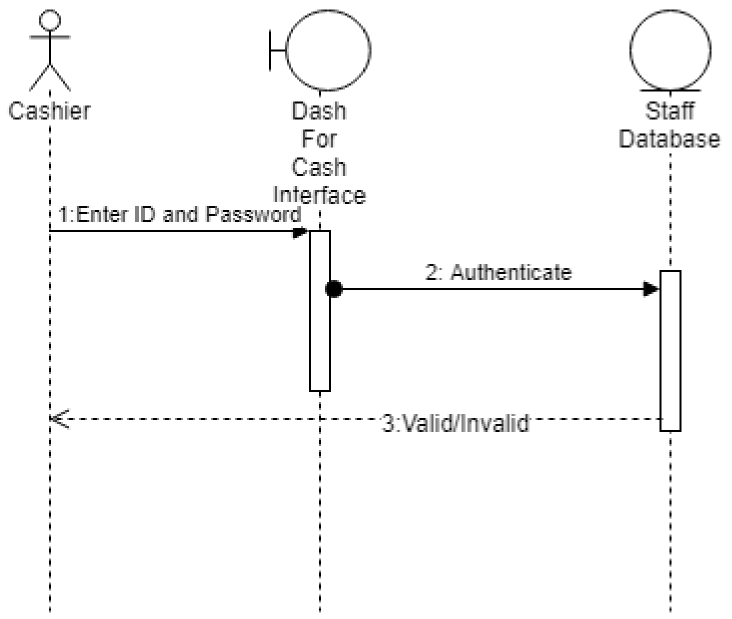
Use Case 9 - Finalize Transaction



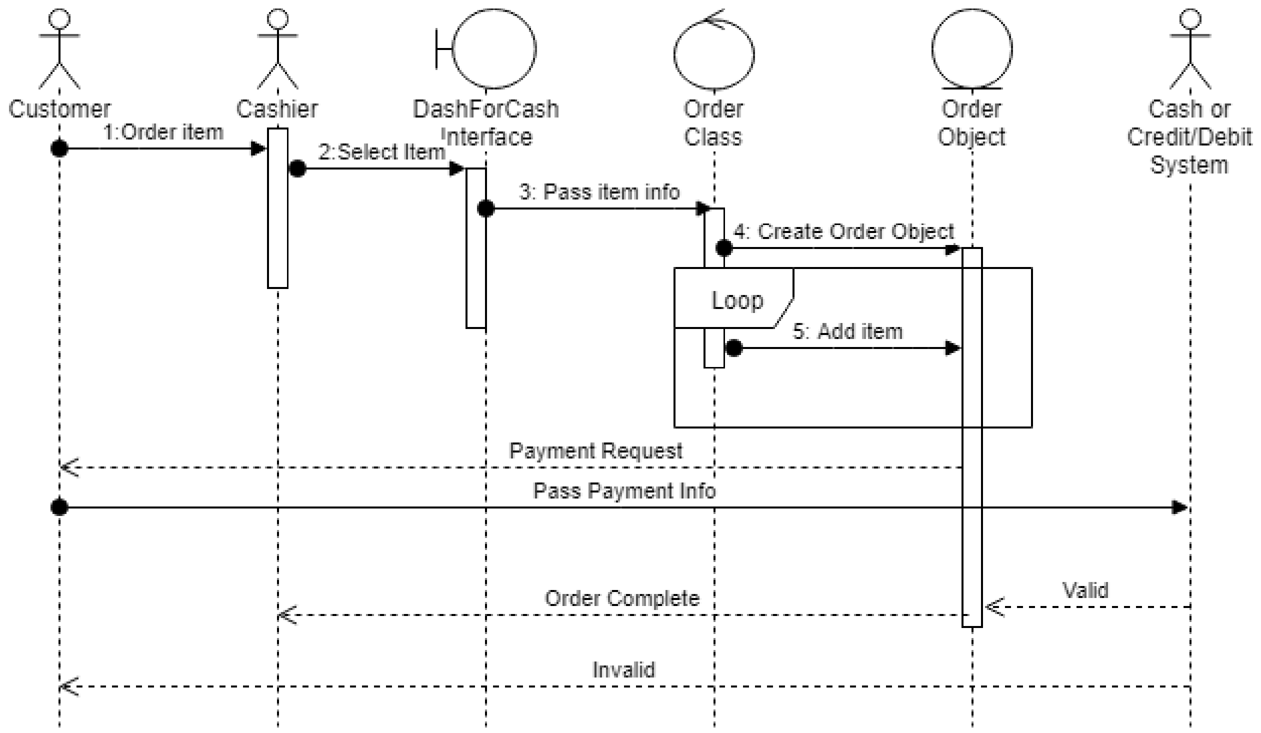
System Sequence Diagrams

Below are shown two common workflows for the Cashier and Customer.

Cashier Login Sequence Diagram



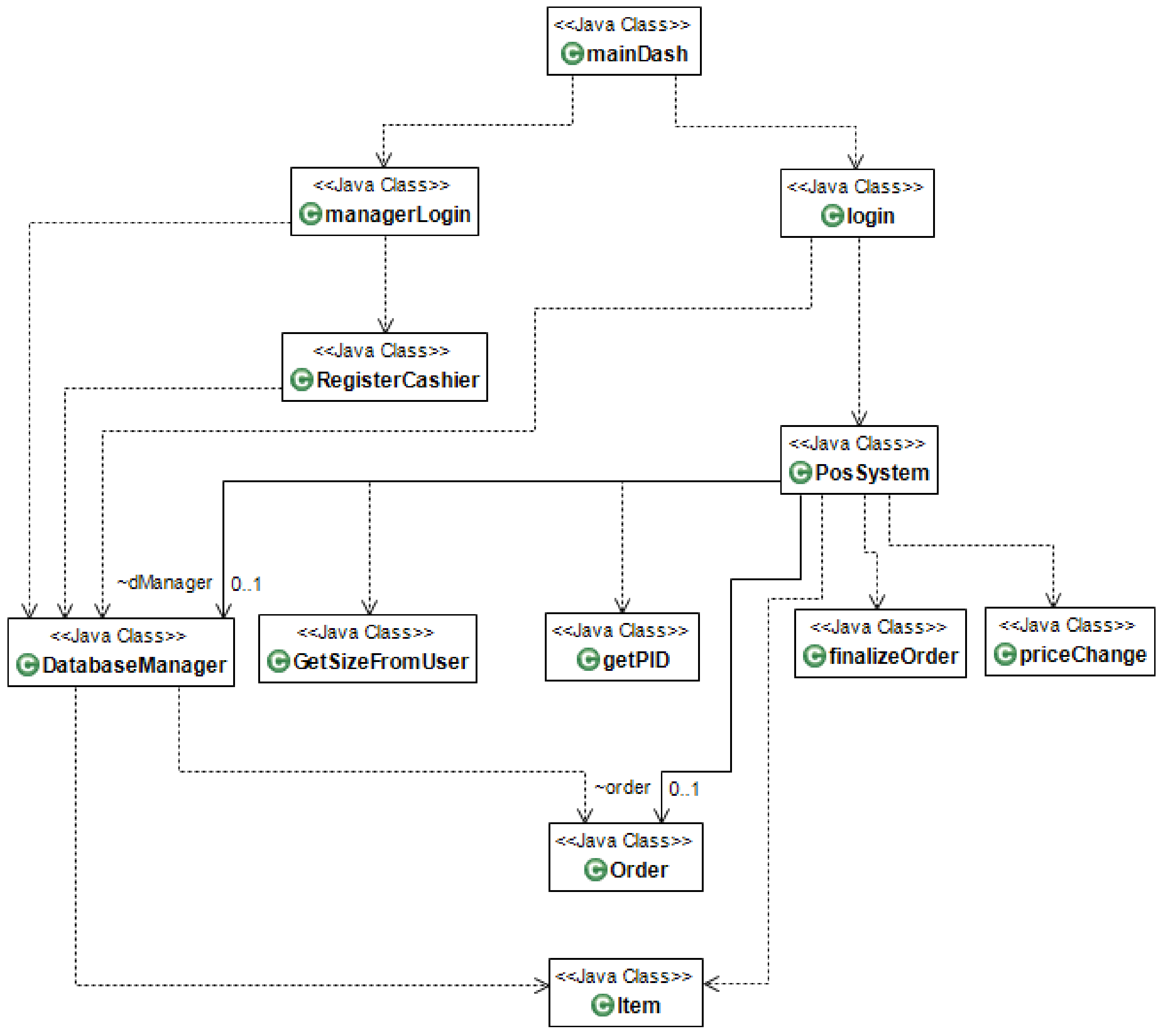
Customer Order



Class Diagram and Interface Specification

Since the class diagram for our system is so large, we instead present a less detailed version. The larger, more detailed version can be found in the zip file (called ClassDiagram).

Less-Detailed Class Diagram



<<Java Class>> mainDash
<<Java Class>> managerLogin
<<Java Class>> login
<<Java Class>> RegisterCashier
<<Java Class>> PosSystem
~dManager
0..1
<<Java Class>> DatabaseManager
<<Java Class>> GetSizeFromUser
<<Java Class>> getPID
<<Java Class>> finalizeOrder
<<Java Class>> priceChange
~order
0..1
<<Java Class>> Order
<<Java Class>> Item

Data Types and Operation Signature

The signatures for the most important classes are shown below:

Item

Fields:

- itemName: String. Stores name of item
- itemPrice: double. Stores price of item
- itemType: int. Stores type of item

Methods:

- getItemPrice(): double. Returns price of item
- getItemName(): String. Returns name of item
- getItemType(): int. Returns type of item
- setItemPrice(double): void. Sets price of item
- setItemName(String): void. Sets name of item
- setItemType(int): void. Sets type of item
- displayItem(): String. Returns description of item including name and price of item

Order

Fields:

- orderId: int. Id of order
- itemID: int. Id of item
- itemList: List<Item>. List of items in virtual cart

Methods:

- addItems(Item, int): void. (Might change)
- removeItem(int): void. Remove item with certain id from cart
- changePrice(Item, double, int, int): void. Change price of item in cart
- returnOrder() Item[]. Return itemList as an array
- getTotal() double. Return total price of items in cart

RegisterCashier

Fields:

- cashierReg: DatabaseManager

Methods:

- jButton1ActionPerformed(java.awt.event.ActionEvent): void. Registers cashier when user presses appropriate button.
- jButton2ActionPerformed(java.awt.event.ActionEvent): void. ????

managerLogin

Fields:
- employeeId: int. Employee ID
- Password: String. Employee password

Methods:
- jButton2ActionPerformed(java.awt.event.ActionEvent): void. ???
- jButton1ActionPerformed(java.awt.event.ActionEvent): void. Allows user to login if provided user ID and password belong to manager in database

login

Fields:
- employeeId: int. Employee ID
- password: String. Employee password

Methods:
- jButton2ActionPerformed(java.awt.event.ActionEvent): void. Check if employer ID and Password are stored in database
- getEmployeeId(): int. Return employee Id
- getPassword(): String. Return employee password

DatabaseManager

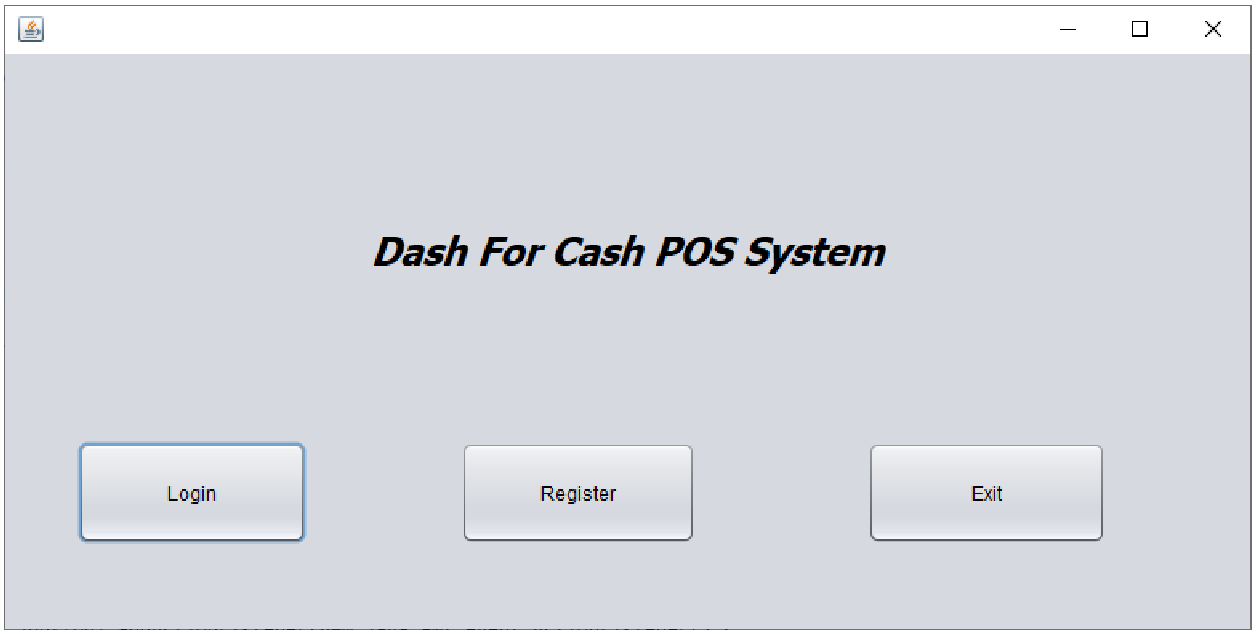
Fields:
- DB_URL: String. Address of database
- DB_DRV: String. Location of JDBC driver
- DB_USER: String. Manager user ID
- DB_PASSWD: String. Manager password
- connection: Connection: Represents connection with specific database

Methods:
- setCon: void. Creates connection with a database
- conClose: void. Closes connection with a database
- registerCashier(String, String, String): int. Talks to database to register cashier
- authenticate(int, String): int. Talks to database to check user credentials
- retrieveOrder(int): Order. Retrieve price of item from database
- retrieveItem(int): Item. Returns requested rows from database
- saveOrder(Object): int. Saves transaction in database (?)

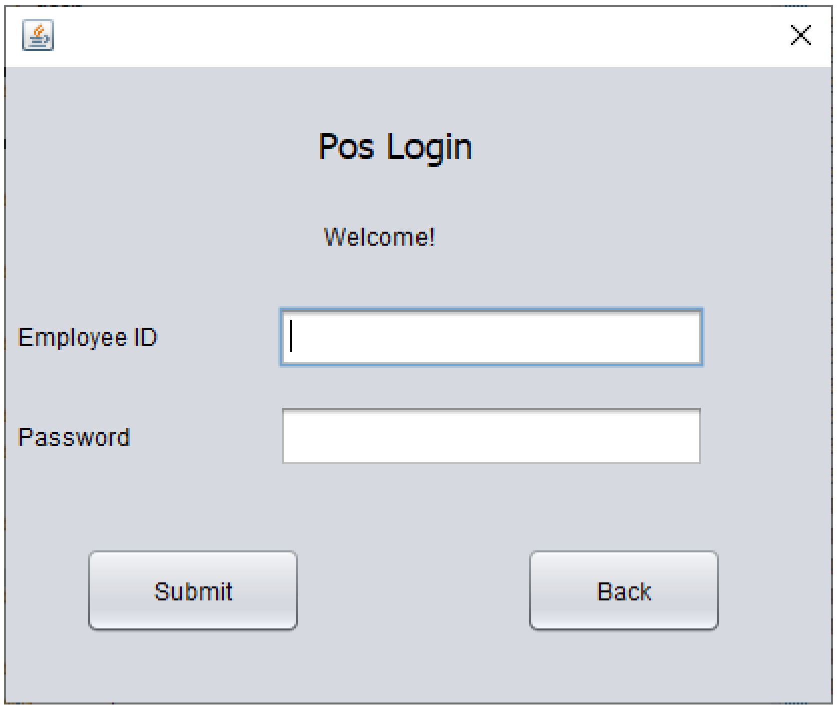
User Interface Design and Implementation

The most important screens from our application are shown below.

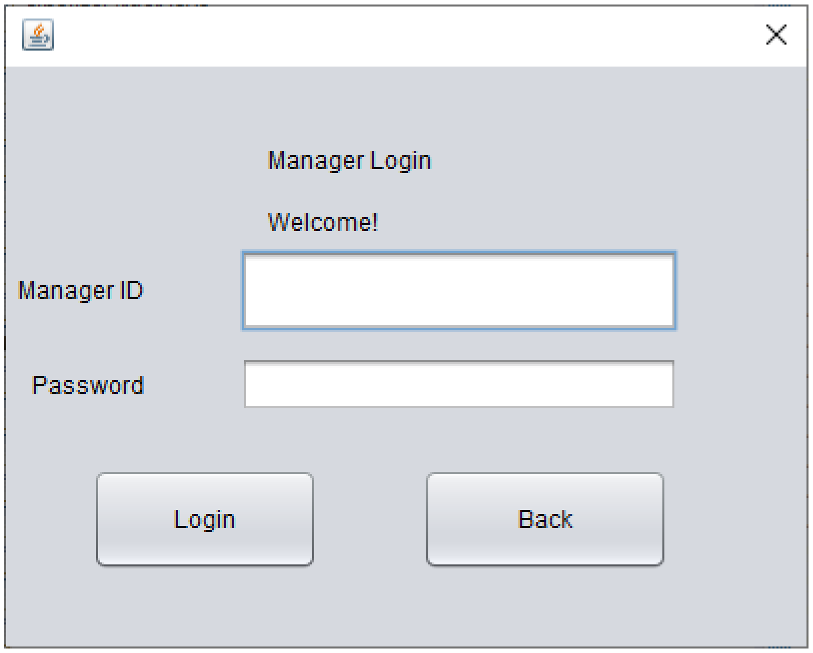
Startup Screen



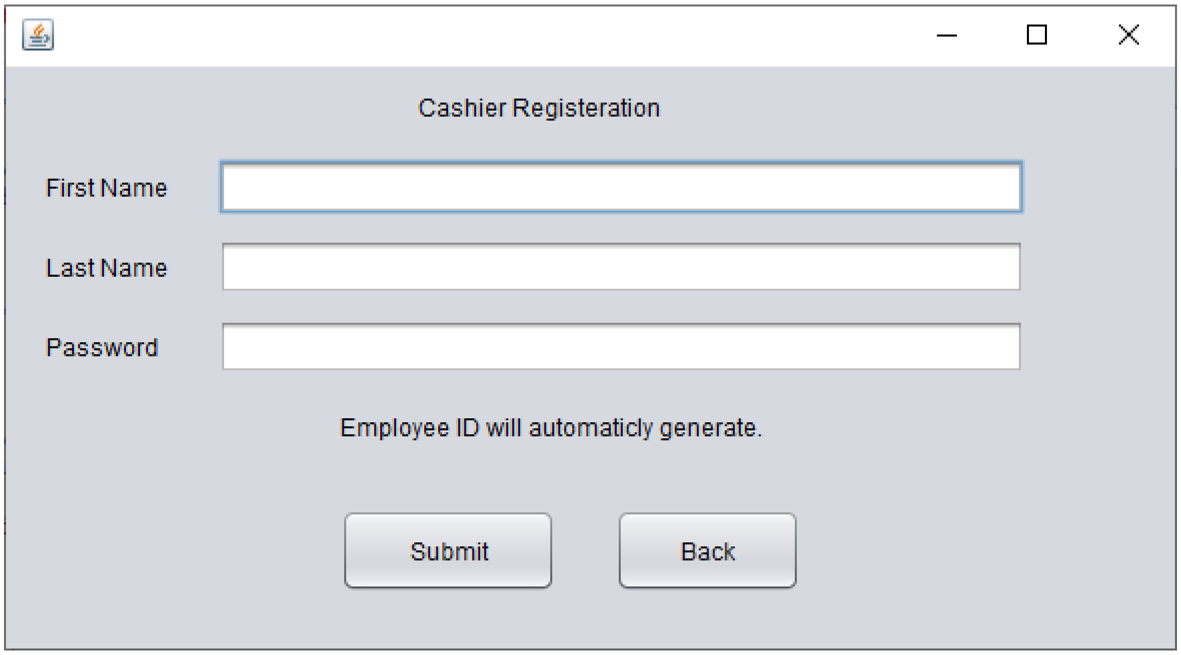
User Login Screen



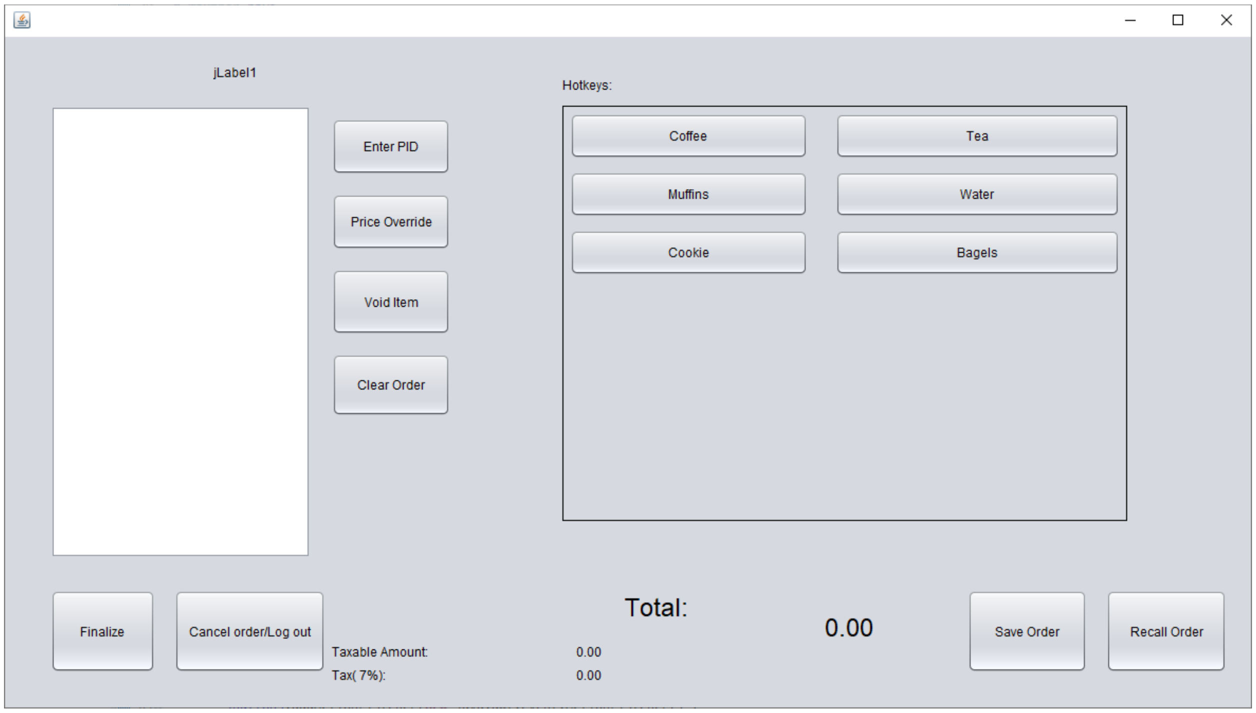
Manager Login Screen



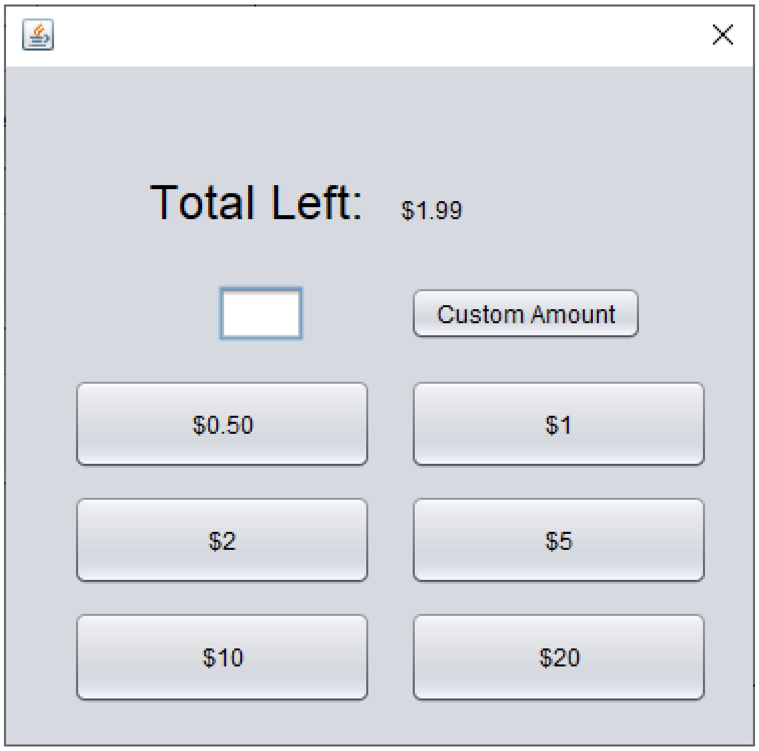
Cashier Registration Screen



Cashier’s Virtual Cart Screen



Finalize Transaction Screen



Database Design

The schema for the database is shown below. The database stores information that all clients need to access, and stores the following three entities: Employees, Orders, and Items.

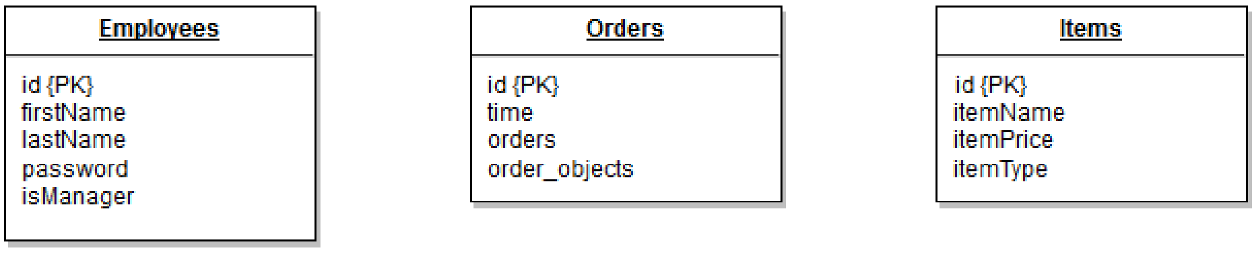
The Order entity contains information relating to saved transactions. The information stored in this entity allows cashiers to restore previously saved transactions.

The Items entity maps all items sold by the store to a price, size (if applicable), and unique ID.

The Employees entity stores the first name, last name, and password of all employees. It also records whether that employee is a manager. This is important since only managers can add employees to the database.

As for the programming itself, we followed Debnath's advice and used JDBC to get Java to interact with the MySQL database (Debnath, 2017).

Database Schema



Notice that this schema does not contain any relationships. This is not a mistake! Relationships simply were not needed for this design. The notation {PK} refers to the Primary Key of that entity.

Design of Tests

Because we were tight on time, and because our classes are so simple, the unit testing was replaced with careful examination of the code.

Component testing was done by first testing the common workflows of each user. We then tested the edge cases by examining more unexpected scenarios. We admit that a few of the failed test cases could not be fixed on time.

Test ID: Login

Assumptions: The general login screen is displayed

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**
Valid UserID, valid password	Login Successfully	Pass		Verifies that an employee can login to system
Invalid UserID, valid password	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-employees are not able to login to system

Valid UserID, invalid password	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-employees are not able to login to system
Invalid UserID, invalid password	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-employees are not able to login to system

Test ID: Manager Login

Assumptions: The manager login screen is displayed

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**
Valid UserID, valid password	Login Successfully	Pass		Verifies that manager can get to management screen
Do not enter anything. Submit.	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-managers cannot get to management screen
Invalid UserID, valid password	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-managers cannot get to management screen
Valid UserID, invalid password	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-managers cannot get to management screen
Invalid UserID, invalid password	Not able to login. User told they have incorrect credentials	Pass		Verifies that non-managers cannot get to management screen

Test ID: Add Cashier To Database

Assumptions: Manager is user, and ready to make request.

Test Description:

WORKFLOW	EXPECTED RESULTS	PASS/FAIL	REASON FOR FAILURE	COMMENTS
Cashier not in database	New data is created in database. User is given confirmation.	Pass		Verifies that new cashiers can be added to database
Cashier already in database	New data is created in database. User is given confirmation.	Pass		Verifies that new cashiers can be added to database
Create cashier without filling in all details	If password is not given, database should be unchanged. Return error.	Pass		Verifies that users in database must have password. Otherwise, someone who knows just username can log in.

Test ID: Add To Cart

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	EXPECTED RESULTS	PASS/FAIL	REASON FOR FAILURE	COMMENTS
Add item by selecting category icon	Item added to cart. Total Price is updated.	Pass		Verifies that cashier can add items to user cart using category icons
Add item by entering valid PID	Item added to cart. Total Price is updated.	Basically Pass	If X is clicked to avoid entering PID, an item with price $0 is added to cart. Not huge deal, just odd	Verifies that cashier can add items to user cart using PID
Add item by entering invalid PID	Cart unchanged. Warning shown to cashier, warning that PID is invalid	Basically Pass	An item with price $0 is added to cart. Not huge deal, just odd	Verifies that PID's that do not correspond to any product in the database are handled properly

Test ID: Remove from Cart

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	EXPECTED RESULTS	PASS/FAIL	REASON FOR FAILURE	COMMENTS
Remove item from cart by selecting item on in cart and clicking Void Item	Item is removed from left-hand side. Total Price is updated.	Pass		Verifies that cashier can remove items from virtual cart
Click Void Item without first selecting item in cart	No action should be taken	Pass		Verifies that virtual cart is left unchanged when cashier attempts to remove unspecified item

Test ID: Clearing Cart

Assumptions: The screen is displaying the virtual cart is shown

Test Description:

WORKFLOW	EXPECTED RESULTS	PASS/FAIL	REASON FOR FAILURE	COMMENTS
Items in cart	All items are removed from left-hand side. Total Price is set to 0.	Fail	Depending on the system, this may pass or fail	Verifies that cashier can remove all items from virtual cart
No items in cart	No action should be taken	Pass		Verifies that virtual cart is left unchanged when cashier attempts to remove 0 items

Test ID: Change Quantity in Cart

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	EXPECTED RESULTS	PASS/FAIL	REASON FOR FAILURE	COMMENTS

Add same item multiple times	Each instance of item should appear in cart. Final Price updated	Pass		Verifies that customer is able to purchase items in varying quantities

Test ID: Override Price

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**
Select item from cart and enter valid price	Price of item in cart in changed. Total Price is updated.	Basically Pass	Selecting X or back to cancel out changes price of item to $0.	Verifies that cashier can override the price of item
Select item from cart and enter invalid price	Price of item in cart is unchanged. Total Price is unchanged. Warning is given to user, asking for a valid input	Basically Pass	Adds an item with price $0 to cart.	Verifies that cashier cannot assign an invalid price to an item in customer cart.
Select Price Override before selecting an item in customer cart.	Nothing should happen.	Pass		Verifies that cashier can only change the price of an item in customer cart.

Test ID: Transactions

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**
Items in customer cart. Transaction is Saved.	Message should be displayed to user confirming transaction has been saved. Transaction saved in Database. Empty customer cart is displayed.	Fail	Total Price is set to 0. Add new item and total price is incorrect.	Verifies that cashier can save a transaction.

No items in customer cart. Transaction is Saved.	Database unchanged. Message displayed to user asking to first add items (no popup needed)	Pass		Verifies that cashier cannot save an empty transaction.
No items in customer cart. Transaction is Retrieved.	Items in transaction appear in virtual cart. Total Price is updated.	Pass		Verifies that cashier can retrieve a transaction.
Items in customer cart. Transaction is Retrieved.	Previous items in cart are *replaced* by new items. Total Price is updated.	Pass		Verifies that virtual carts only ever contain the items for one customer.

Test ID: Cancel Transaction

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**
No items in customer cart, and Cancel Transaction is selected	User is brought back to login screen	Pass		Verifies that after cancelling transaction without any items in cart does not do something unexpected.
Items in customer cart, and Cancel Transaction is selected.	User is brought back to login screen	Pass		Verifies that after cancelling transaction, previous items no longer in cart.

Test ID: Finalize Transaction Button

Assumptions: The screen is displaying the virtual cart

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**

At least one item in customer cart	New screen allowing cashier to display price is shown	Pass		Verifies that cashier can begin to finish transaction
No items in customer cart	Message displayed to cashier to first add items.	Pass		Verifies that customer cannot pay nothing

Test ID: Finish Transaction

Assumptions: The Finalize Transaction button has just been selected

Test Description:

WORKFLOW	**EXPECTED RESULTS**	**PASS/FAIL**	**REASON FOR FAILURE**	**COMMENTS**
Select amount to alter price by using buttons	Unknown	Fail	The amount that is pressed is subtracted from final price. Once price goes below 0 though, selected prices add to final price	Verify that cashier can finish transaction by selecting buttons
Select amount to alter price by typing in value	Unknown	Fail	The amount that is pressed is subtracted from final price. Once price goes below 0 though, selected prices add to final price	Verify that cashier can finish transaction typing values in.

Closing Thoughts

This was a challenging project I think for all of us:

- Got off to a slow start and never really recovered
- Poor communication (ex. meetings not scheduled enough in advance, long text-message response time)
- Overly laid-back management style did not provide necessary structure (ex. the informal schedule we had was not used, people given freedom not only on *how* to accomplish things, but *what* even to accomplish)
- People had other responsibilities which made it difficult for them to be fully involved

However, difficult experiences can offer some of the best learning opportunities, and this experience has ingrained *in our very souls* the value of communication and planning.

Finally, we promise that this work is our own, and have tried our best to give credit to our sources of inspiration.

Sincerely,

[illegible]

Singh

[illegible]

Navjot Bhatti, Hardeep Sandhu, and Jonathan Fitz

Notable Tools

The website https://www.draw.io/ was used to create most of the diagrams.

The ObjectAid Eclipse plugin was used to help create the class diagrams.

References

Debnath, M. (2017, May 03). Creating a JDBC Application in NetBeans: A Step-by-Step Guide. Retrieved November 29, 2018, from https://www.developer.com/java/data/creating-a-jdbc-application-in-netbeans-a-step-by-step-guide.html

Sommerville, I. (2016). *Software engineering*. 10th ed. Hoboken: Pearson Higher Eduction, Inc., pp.130-135.