Software Requirements Specification
For
Point of Sales System

Version 1.0

Created By:
Group #2

Jacob Galvez
Jenna Kohlmeyer
Isaiah Garcia
Rodrigo Hernandez

Table of Contents

1. Introduction- Rodrigo

1.1 Purpose of the SRS

The purpose of this document is to introduce and present a detailed description of POS system. It will explain the purpose of the system and features of the system, what the system can do, and how to use the system. This document will also list and describe the statement of functional requirements, non-functional requirements, and design/implementation constraints. This document is intended for presentation purposes and also users' use.

1.2 Problem Statement

In the modern restaurant era, POS systems can be confusing and hard for new employees to understand. With so many restaurants opening and closing every day a free easy to use all in one POS system is what they need. We set out to fill that need with our own POS system version 1. Our system will have multitasking features and will be easy for any restaurant owner to implement in their business.

1.3. Scope

The purpose of POS system is to ease the restaurant going and serving experience. We set out to create an easy-to-use program for restaurant owners so that they can streamline their service to customers and their employees. The system is based on Java UI and will have database capabilities to store orders and check-in times for customers. Above all, our POS system will help any business owner run their operation smoothly, comfortably, and all while at a low cost.

2. Statement of functional requirements - Isaiah

Summary: The point of sales system should allow the user to view and control various aspects of a restaurants operation related to sales, such as price calculations table reservation tracking

1. **Creating an order**: User has a database of items that the restaurant sells, and can select items to calculate the total price of those items.

Pre-Condition: User has successfully navigated to the order screen and started a new order.

Post-Condition: User has a list of all items on order with their prices with a total price of all items presented at the bottom of the list.

1.1. **Use-Case**:

1.1.1. User clicks on “Store Front” button

1.1.2. System opens the table layout, so that you may pick one

1.1.3. User searches and clicks on desired table

1.1.4. System displays the menu in clickable buttons

1.1.5. User selects that item to be added to order

1.1.6. System updates the total price of order

1.1.7. User gets money and closes the order

Welcome page

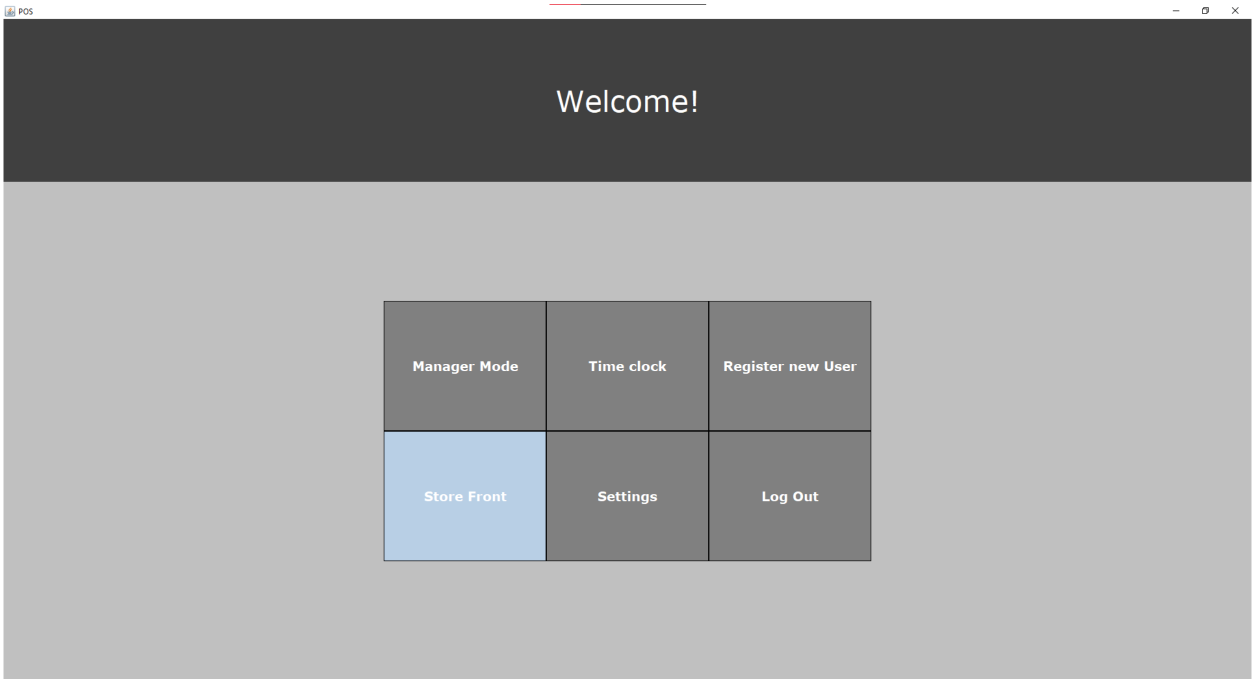
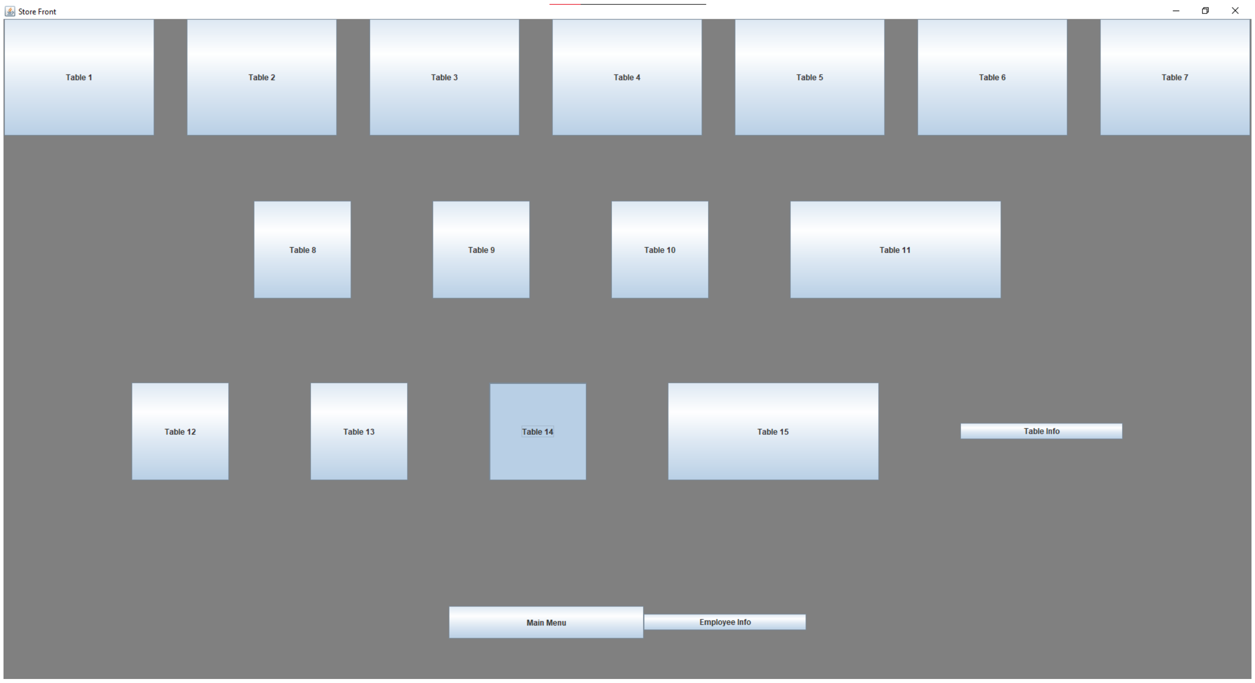
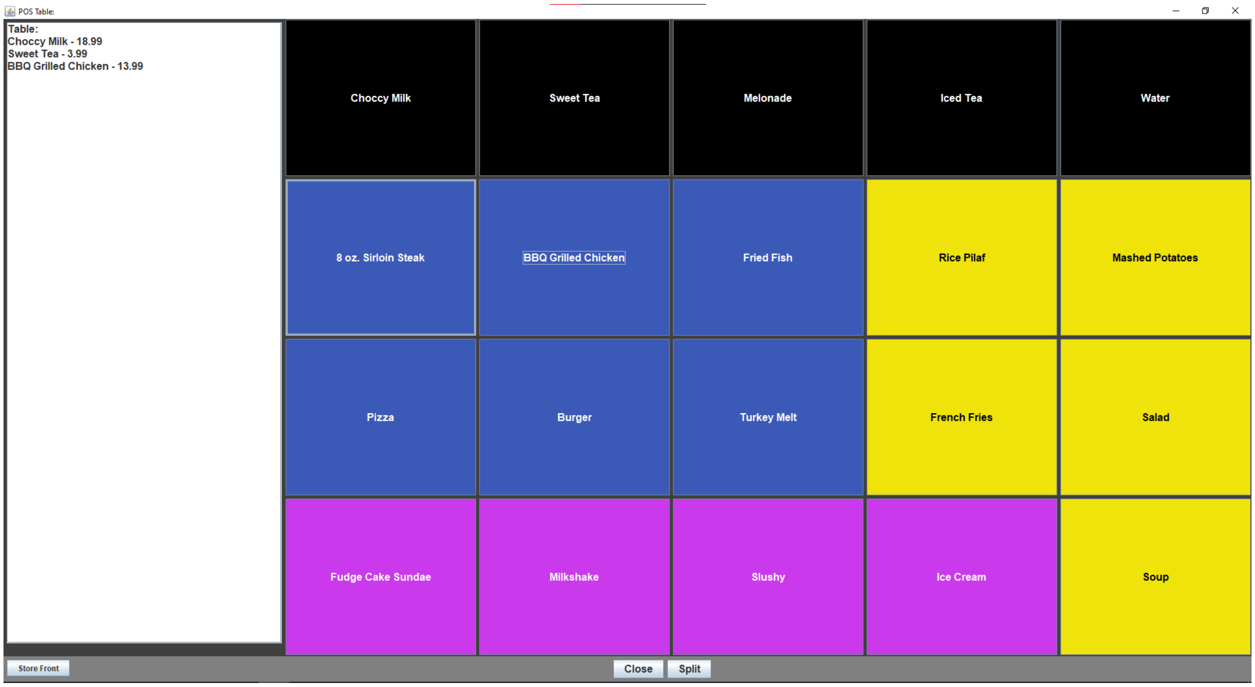


Table Layout GUI

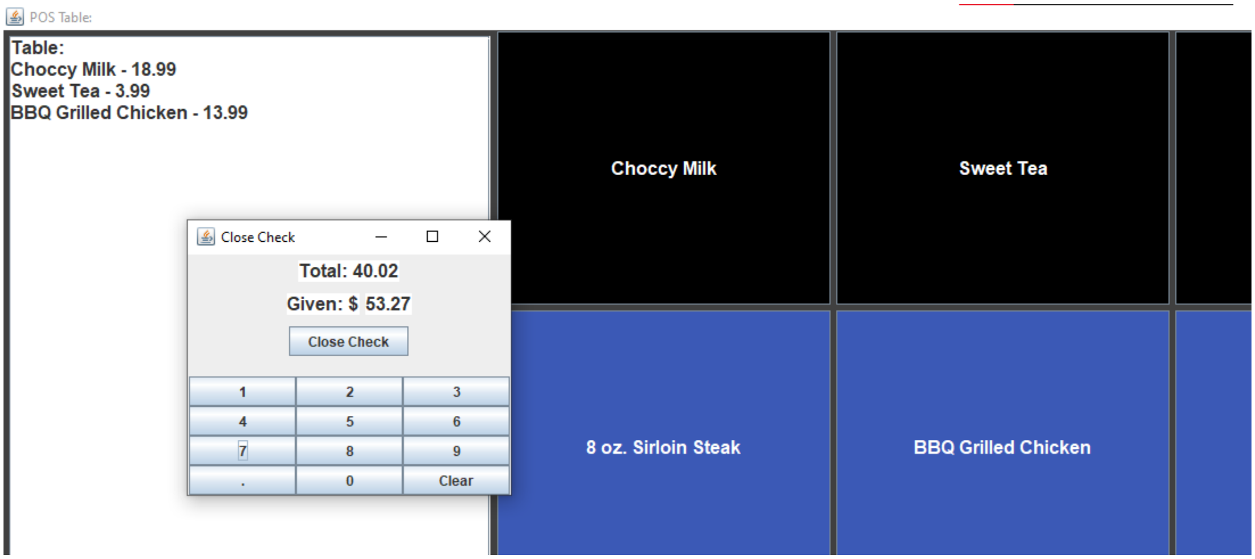


Order Creator



```
Table:
Choccy Milk - 18.99
Sweet Tea - 3.99
BBQ Grilled Chicken - 13.99
```

Change Calculator



```
Table:
Choccy Milk - 18.99
Sweet Tea - 3.99
BBQ Grilled Chicken - 13.99
```

Total: 40.02

Given: $ 53.27

2. **Manager Mode:** Users can change the names of menu items and also are able to change their respective prices as well. There is also a mode where a manager can hire/dismiss new/old employees.

Pre-Condition: User must enter a “Manger Code” (By default set to “258”)

Post-Condition: User has a view of different authorized settings in which they are able to make changes to.

2.1. Use-Case:

2.1.1. User clicks on button to go to Manager Mode

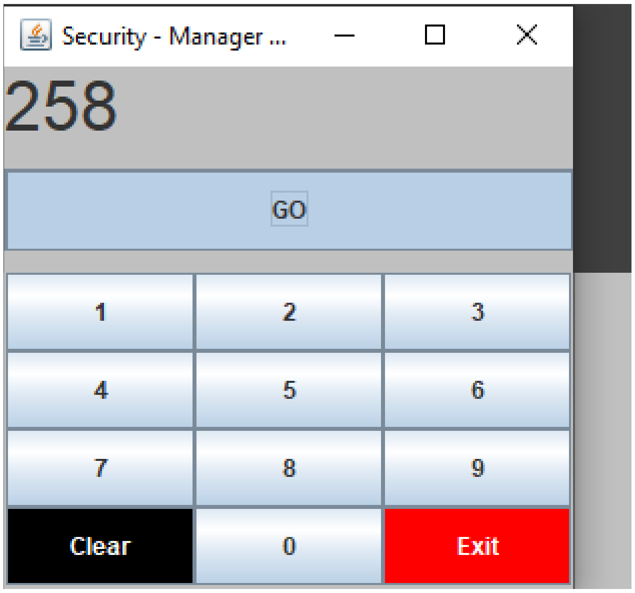
2.1.2. Systems pops up a security code window

2.1.3. User inputs code (“258”)

2.1.4. System prompts gives user access to manager mode

2.1.5. User may change menu items/prices, and fire employees

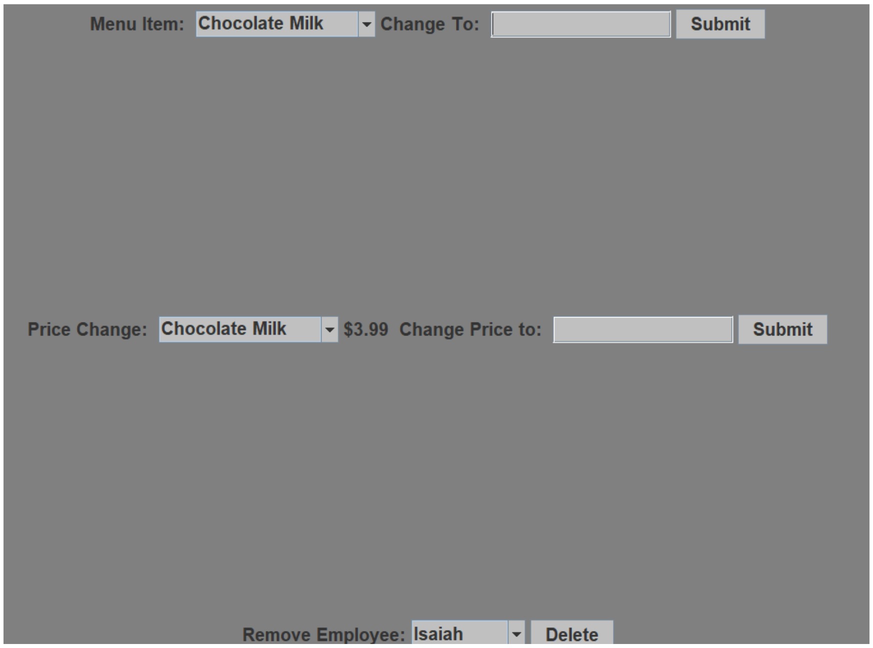
Manager Security Code Lock



Security - Manager ...

258

Manager Mode Options



Menu Item: Chocolate Milk

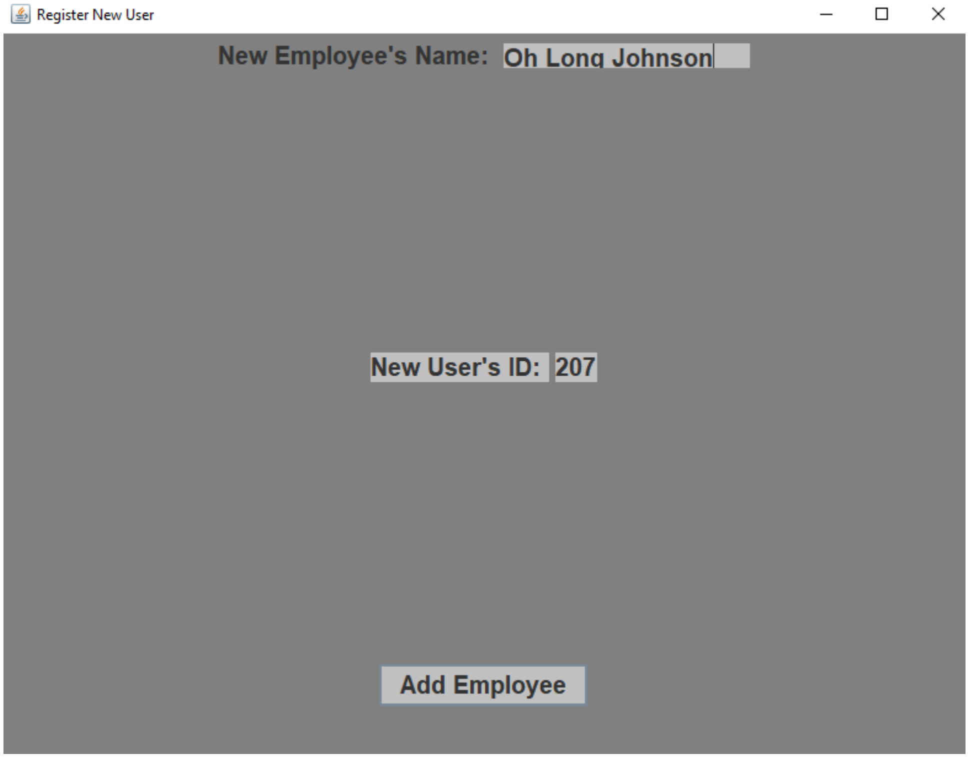
Change To:

Price Change: Chocolate Milk $3.99

Change Price to:

Remove Employee: Isaiah

Register new Employee Form



Register New User

New Employee's Name: Oh Long Johnson

New User's ID: 207

3. **Check Creating/Closing:** User is able to create checks for a table with information on server, items purchased, total price, tax, etc. (Mockup Included with Order Creator Mockup on the Left Side of the screen there is a “check”.) The user just clicks “close” to close the check.

   Pre-Condition: User has navigated to the check creation button.

   Post-Condition: User has created a new check for a table/patron.

   3.1. Use-Case:

   3.1.1. User selects check creation button

   3.1.2. System prompts user to enter table

   3.1.3. User enters table that the check is for

   3.1.4. System automatically enters server information for that table

   3.1.5. User enters the items purchased at that table

   3.1.6. System calculates total cost of items including tax

   3.1.7. System displays a preview of check to be printed

   3.1.8. User confirms that the information is correct and prints the check

4. **Check Splitting:** User can split a check into separate checks based on orders

   Pre-Condition: User has created a check to split.

   Post-Condition: User has access to *n* separate checks from the first

   4.1. Use-Case:

   4.1.1. User finishes creating anew check

   4.1.2. User selects option to split check

4.1.3. System prompts user to enter how many checks to split into

4.1.4. User inputs 2

4.1.5. System presents user with first check to fill out (items, names, etc)

4.1.6. User fills out information on check

4.1.7. System presents user with second check to filll out (items, names, etc)

4.1.8. User fills out information on check

4.1.9. System prompts user to confirm that information on the 2 checks in correct

4.1.10. User confirms and prints out checks

5. **Wait Time Estimation:** User can select a group size and the system will estimate how long that group will have to wait for a table based on available tables and time of each taken table.

Pre-Condition: User is seating new patrons in the restaurant.

Post-Condition: User has been given information on how long the patrons have to wait for their table.

5.1. Use-Case:

5.1.1. User clicks on the “Seat New Patrons” button

5.1.2. System prompts user to enter the number of members in party

5.1.3. User enters the number of members in party

5.1.4. System calculates which table will be empty the soonest and presents the estimated wait time in minutes

5.1.4.1. System calculates estimated wait time by getting tables with enough seats for party, getting the times that each of those tables were seated, subtracts those starting times from the average time to eat for this party size, and presents the table with the soonest ready time to seat new patrons.

5.1.4.2. **Alternative:** If there is a table available for the party size, the system will present the table as open with an estimated wait time of 0 minutes.

6. **Current Staff Information:** System should provide information on staff currently working and where, specifically for servers and which tables they are assigned to.

Pre-Condition: There are staff currently working.

Post-Condition: User is shown information on current server's status and location.

6.1. **Use-Case:**

6.1.1. User clicks on "Tables" button

6.1.2. System shows GUI of the table's current statuses and which staff are assigned to which tables.

7. **Clock In/Out:** User is able to select their username and clock in when starting their shift and clock out when they are finished with their shift.

Pre-Condition: User is a current employee of the establishment and has a username to clock in with.

Post-Condition: User has logged the time they started working and time they stopped working with the system.

7.1. **Use-Case:**

7.1.1. User clicks on Employee button

7.1.2. System presents the user with a list of employees

7.1.3. User selects their name from the list of employees

7.1.3.1. **Optional:** User enters a unique passcode to validate

7.1.4. System presents option to clock in or clock out

7.1.5. Scenario A: User selects the clock in button

7.1.5.1. System logs the time button was clicked and starts tracking time passed

7.1.6. Scenario B: User selects the clock out button

7.1.6.1. System logs the time the button was clicked and stores amount of time since user clocked in.

8. Combine Tables: User is able to select 2 tables within a vicinity in the Tables GUI and combine them into 1 table for larger parties.

Pre-Condition: User has navigated to the Tables GUI

Post-Condition: System treats and displays 2 tables as 1 combined table

8.1. Use Case:

8.1.1. User selects the Table GUI button

8.1.2. System displays the Table GUI, showing all tables and their status

8.1.3. User selects the combine table button

8.1.4. System fades out all the tables

8.1.5. System asks user to select the first table

8.1.6. User selects the table they wish to start with

8.1.6.1. Note: To be eligible to combine, the first table must be free from patrons and not be reserved for a certain amount of time in the future.

8.1.7. System highlights that table and highlights all nearby tables that are eligible to be combined

8.1.7.1. **More:** To be eligible to combine tables, the second table must be within a certain distance of the first table, must be free from patrons, must not be reserved for a certain amount of time in the future, and must be a free-standing table (i.e. not a booth fixed to the wall)

8.1.8. User selects an eligible table to combine

8.1.9. System gives both tables a visual to indicate that they are being treated as one table

8.1.10. Both tables are now treated as one table when creating checks, assigning servers, and other table operations.

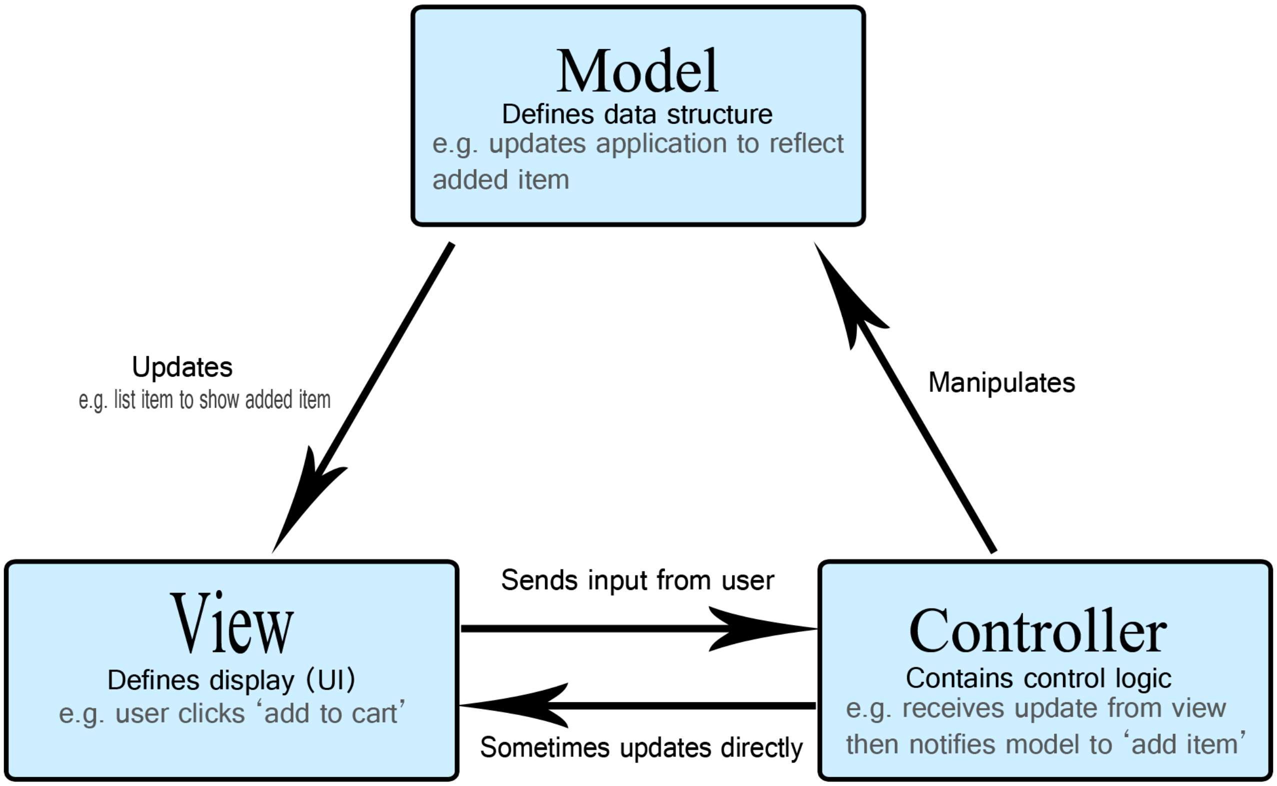
3. Non-functional requirements

- Performance requirements
    - The program should be able to perform all day, with no downtime.
    - The program must be very responsive and must be able to keep up with a fast-paced work environment. Most functions should be able to
    - The program must display errors when an error occurs, the user should be notified of what they are doing wrong, etc…
    - When the program is shut down, all of the data is saved onto a file for bookkeeping.
    - Timeout window period. If the program has not been used for 45 seconds, the user is logged out.
- Usability requirements
    - The point of sales system must have an easy-to-use UI and should be easy to pick up for new people to learn how to use. The program should be very clear and concise on what functions you are trying to do on the POS.

- Reliability requirements (e.g. system availability, error handling)
    - In the event that an error does occur, the system must let the user know the error that is causing this problem, and what they need to do to fix it. Also, again the POS should be able to run all day (around 16 hours) with no downtime. It must be able to run at all hours that the business is open and selling items.

4. Design/implementation constraints

- Hardware Constraints
    - The system should be designed to run on standard hardware configurations such as desktops and laptops, with a minimum of 4GB of RAM and a dual-core processor.
    - No card reader, cash only
- Software Constraints
    - The system should be developed using Java programming language and should be compatible with operating systems such as Windows, macOS, and Linux.
    - We will also follow an MVC design pattern for this project. As Shown in the Figure. This will allow the program to become more flexible, efficient, and maintainable.


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

- Security Constraints
  - The system should be designed to be intuitive and user-friendly, with a user interface that is easy to navigate and understand.
  - The system should provide clear feedback to the user, such as confirmation messages, and error messages, to improve the user experience.
- Legal Constraints

  - The system should comply with the most relevant laws and regulations (San Marcos, TX), including tax regulations.
  - It may take more time than we are given to implement all laws and regulations including data protection laws, and protection from fraudulent activity, to ensure compliance with financial regulations

5. References and sources of information

SRS example information

6. Glossary/Definitions

POS - Point of Sales

MVC - Model View Controller

Down Time - Time accumulated from a program that is unable to run for whatever reason.