# Software Requirements Specification

for

# 11. Inventory Management and Billing System

**Version 0.1**

**Prepared by**

| | |
|---|---|
| **Student Roll Number: M210662CA** | **Student Name: Junaid Ansari** |
| **Student Roll Number: M210695CA** | **Student Name: Arham Siddiki** |
| **Student Roll Number: M210687CA** | **Student Name: Karan Sachin Kumar** |

**Project Owner:** Akhila

**Course:** CS4097D Object Oriented Systems Laboratory

**Date:** 07.11.2022

# 1 Introduction

A simple grocery shop desktop application where they keep track of stock available and make bills for an order. This is a management system and limited to store only and is not available for general public.

## 1.1 Document Purpose

This SRS describes the software functional and non-functional requirements for release 0.1 of the Inventory Management and Billing System (IMBS). This document is intended to be used by the members of the project team that will implement and verify the correct functioning of the system. Unless otherwise noted, all requirements specified here are high priority and committed for release 0.1.

## 1.2 Product Scope

This system allow its user(point of sales personnel) to view the existing stock with the shop. Further it allows them to add new items to the stock, modify the existing stock whether new stock being introduced or existing stock being declined, remove any item that is no longer served by shop.
In addition, this system also provides them an interface to prepare bills for corresponding items of an order. The shop will also keep a downloaded record of bills in order to cope the future disputes.

## 1.3 Definitions, Acronyms and Abbreviations

IMBS : Inventory Management and Billing System
POS : Point of Sale
DBMS : DataBase Management System
DB : DataBase

## 1.4 Document Conventions

### Formatting Convention

1. Times font style of size ‘12’ is used throughout the document. For heading font size of 30, 18, 14, 12 is used.
2. Italics is used for comments.
3. Bold is used for Headings.
4. 1’’ margin is maintained throughout the document.
5. Lists are represented using numbered list and bullet list.

### Naming Convention

1. Heading are in title case all starting with capital letter and rest small letters.
2. Functions calls are written in camelCase.

## 1.5 References and Acknowledgments

1. My Code Space youtube channel for tutorials.
   www.youtube.com/c/MyCodeSpace1

# 2 Overall Description

## 2.1 Product Overview

IMBS aims to replace manual record keeping of stock along with manual raw billing system. This enables the proprietor to have extended degree of control on stock and sales, along with transparent accounting and thereby restricting the scope of window dressing. It also makes the functioning of shop smooth since availability of any item can be checked instantly without needing to head to inventory. Also providing customer with bills helps both parties(customer and shop) to settle their disputes and claims well.

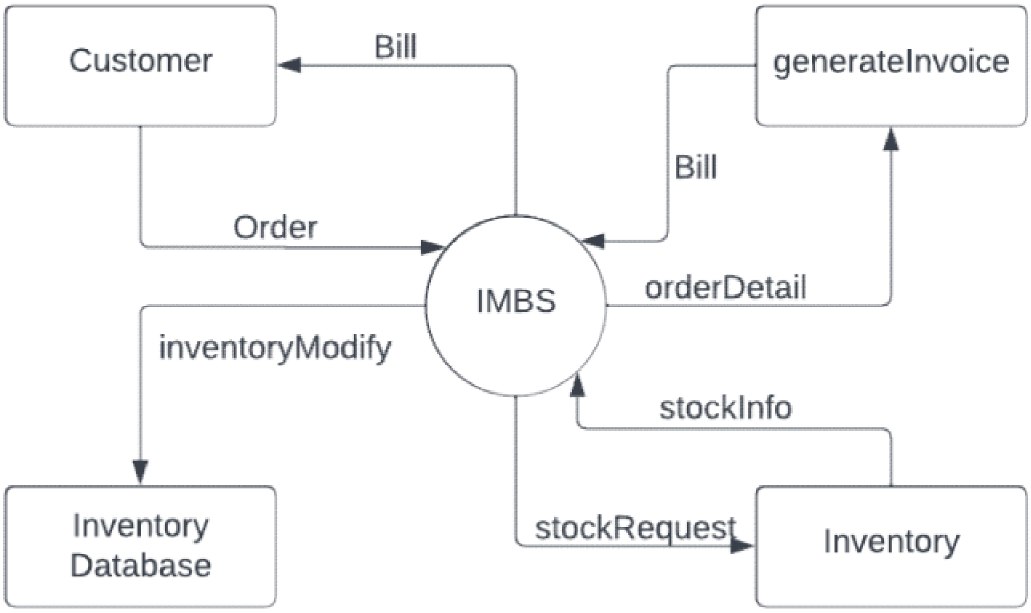


Fig 1. - General context diagram for IMBS

## 2.2 Product Functionality

- Login
- Take orders.
- Generate invoice.
- View inventory.
- Add items to inventory.
- Remove items from inventory.
- Alter current details of an item.
- Search for an item in stock.
- Manage POS.

# 3 System Specification

## 3.1 Functional Requirements

**F1 :** The system shall shows a list of available options such as generate invoice, view inventory, search item, add item, etc.

**F2 :** The system shall take order details and prepare the corresponding invoice for order.

**F3 :** The system shall show the inventory to the user that contains item name, item code, price and available stock.

**F4 :** The system shall be able to add a new item to existing inventory provided with item name, price and stock.

**F5 :** The system shall be able to search an item given code, from the inventory and show corresponding stock available.

**F6 :** The system shall remove an item from inventory given the item's code.

**F7 :** The system shall modify the current details of any item in inventory.

**F8 :** The system shall be able to manage POSs when logged in as admin.

## 3.2 Use Case Model

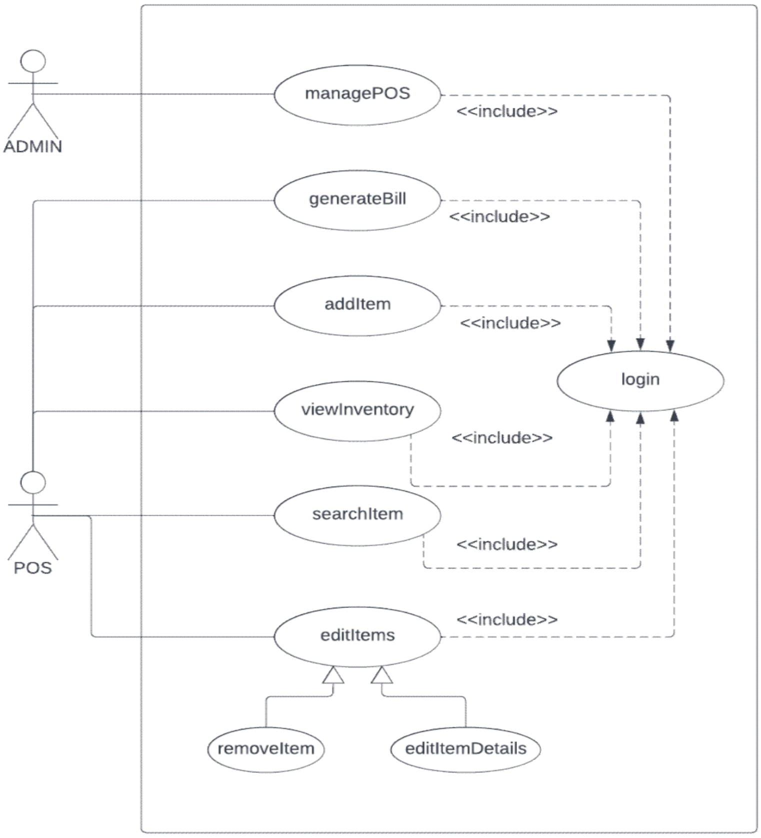


Fig 2. - Use Case diagram for IMBS

### 3.2.1 Use Case #1 (managePOS)

**Purpose** - This use case is designed so the user can manage POS i.e. add or remove a POS. Also he can view total sales made my them.

**Requirements Traceability –** F8.

**Actors** – Admin.

### 3.2.2 Use Case #2 (generateBill)

**Purpose** - This use case is designed so as it can take order from customer for numerous items they want and fetch their corresponding prices from DB and calculate the total amount payable by that customer.

**Requirements Traceability –** F1, F2, F5, F7.

**Actors** – POS personnel(generally sales assistant).

### 3.2.3 Use Case #3 (addItem)

**Purpose** - This use case is designed so as it can add a new item to existing inventory. It takes inputs as item name, item code, item price and stock(in units).

**Requirements Traceability –** F1, F4, F5.

**Actors** – POS personnel(generally sales assistant).

### 3.2.4 Use Case #4 (viewInventory)

**Purpose** - This use case is designed so as it can show the user current inventory i.e. the items which are in stock currently. It consists of tabular representation with columns as S.No , Item name, Item code, Item price, Current stock.

**Requirements Traceability –** F1, F3.

**Actors** – POS personnel(generally sales assistant).

### 3.2.5 Use Case #5 (searchItem)

**Purpose** - This use case is designed so the user can searhc for an item whether it's available in stock or not.

**Requirements Traceability –** F1, F3.

**Actors** – POS personnel(generally sales assistant).

### 3.2.6 Use Case #6 (editItems)

**Purpose** - This use case is designed so as it can make modifications in current stock of items. It serves two purposes as :

1. removeItem : This shall remove item from current inventory and all of its corresponding data from DB and inventory will be removed.
2. editStock : This shall edit the existing stock of an item(increment/decrement).

**Requirements Traceability –** F1, F5, F6, F7.

**Actors** – POS personnel(generally sales assistant).

This page is intentionally left blank

# Design Document

**for**

# 12. Inventory Management and Billing System

Version 0.1

Prepared by Team 10:

| | |
|---|---|
| **Student Roll Number: M210662CA** | **Student Name: Junaid Ansari** |
| **Student Roll Number: M210695CA** | **Student Name: Arham Siddiki** |
| **Student Roll Number: M210687CA** | **Student Name: Karan Sachin Kumar** |

(Based on SRS Version 0.1 prepared by Team 10)

| | | |
|---|---|---|
| **Junaid Ansari** | **M210662CA** | **junaid_m210662ca@nitc.ac.in** |
| **Arham Siddiki** | **M210695CA** | **arham_m210695ca@nitc.ac.in** |
| **Karan Sachin Kumar** | **M210687CA** | **karan_m210687ca@nitc.ac.in** |

| | |
|---|---|
| **Project Owner:** | **Akhila** |
| **Course:** | **CS4096 Software Engineering Laboratory** |
| **Date:** | **14.11.2022** |

***November 14, 2022***

# Glossary

| | |
|---|---|
| IMBS | Inventory Management and Billing System |
| POS | Point Of Sale |
| DBMS | DataBase Management System |
| DB | DataBase |

# Table of contents

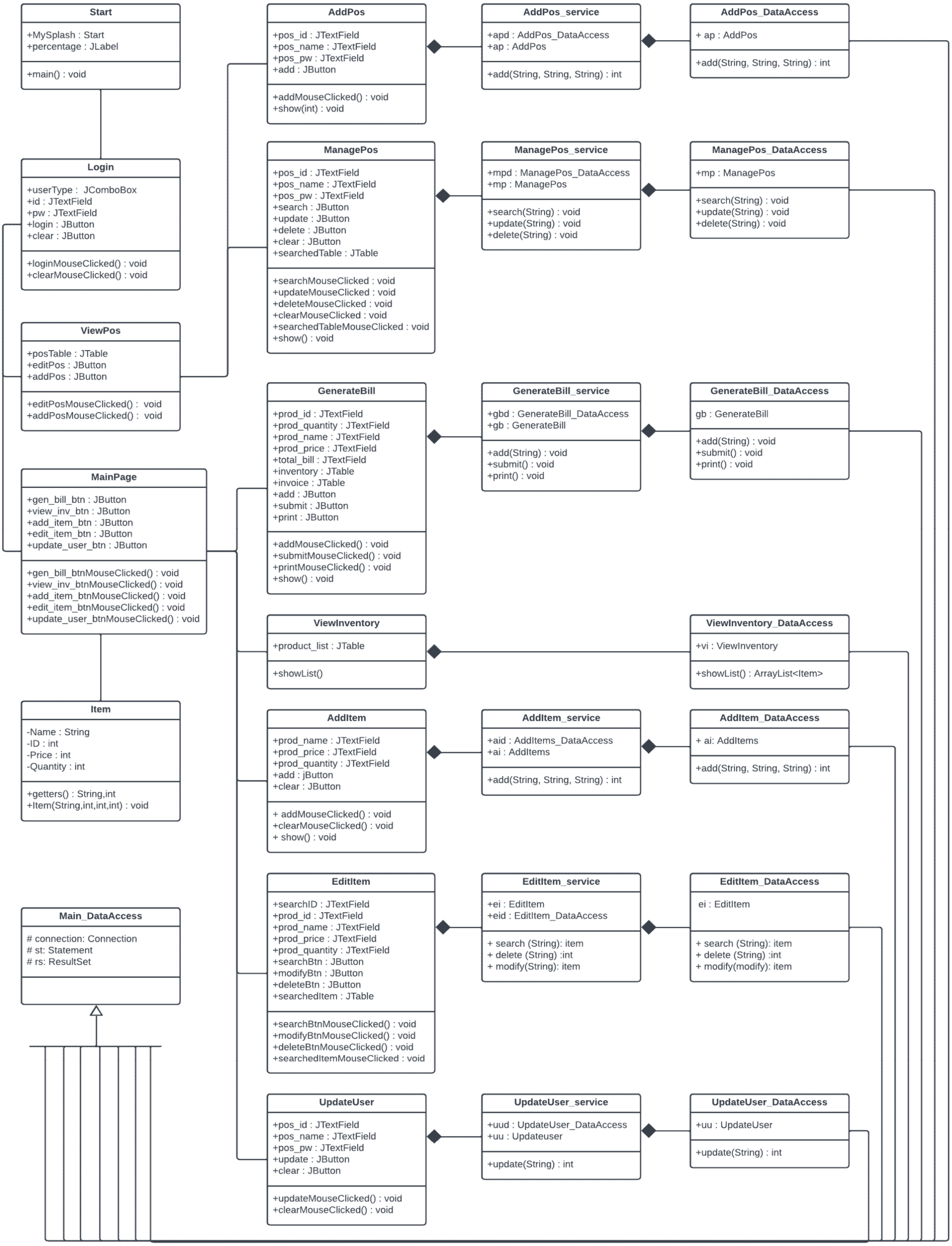
# 1. Detailed Design through UML diagrams

## 1.1 System model using Class Diagram

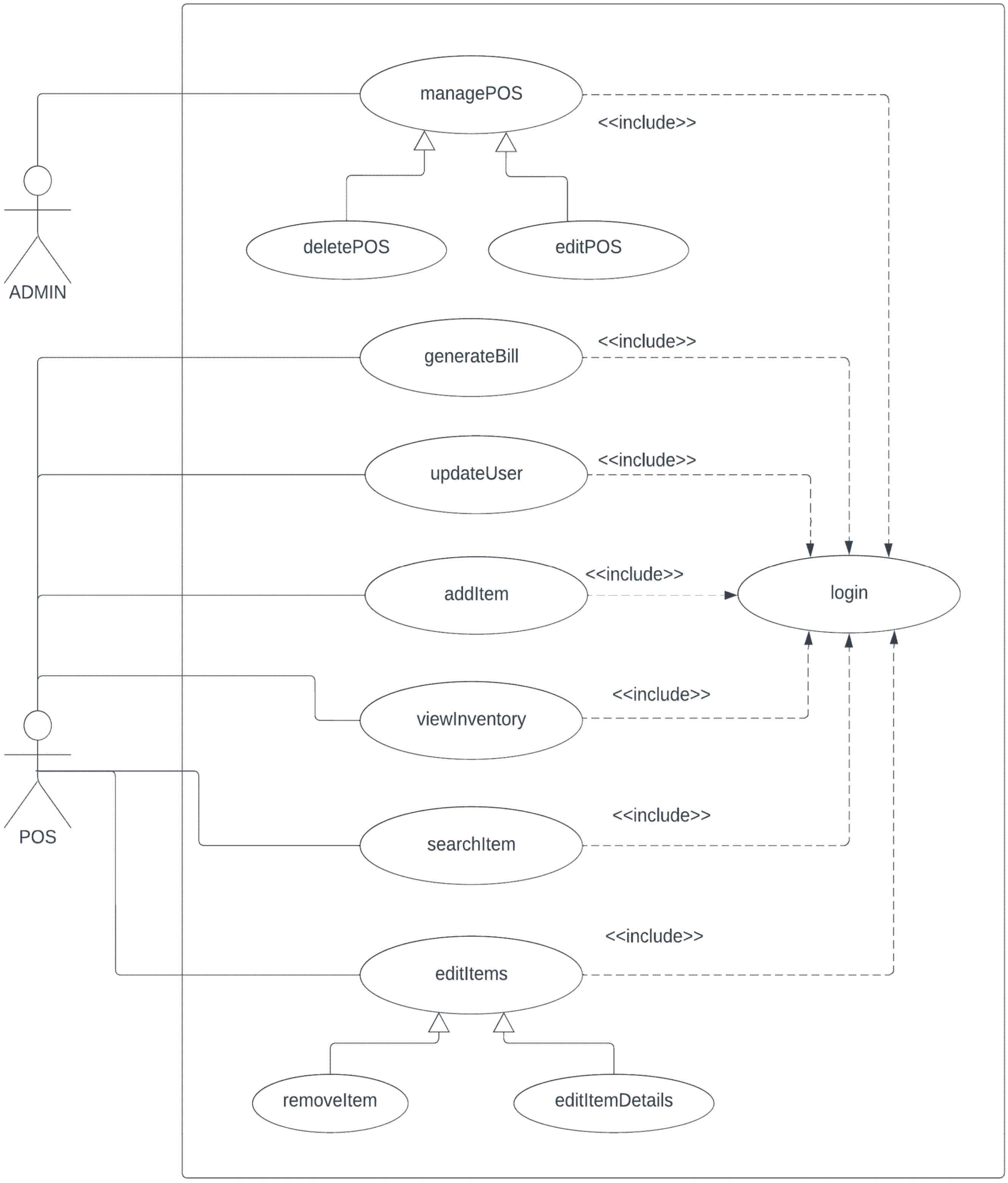
Class Diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods) and the relationships among classes.

//class diagram is on next page//

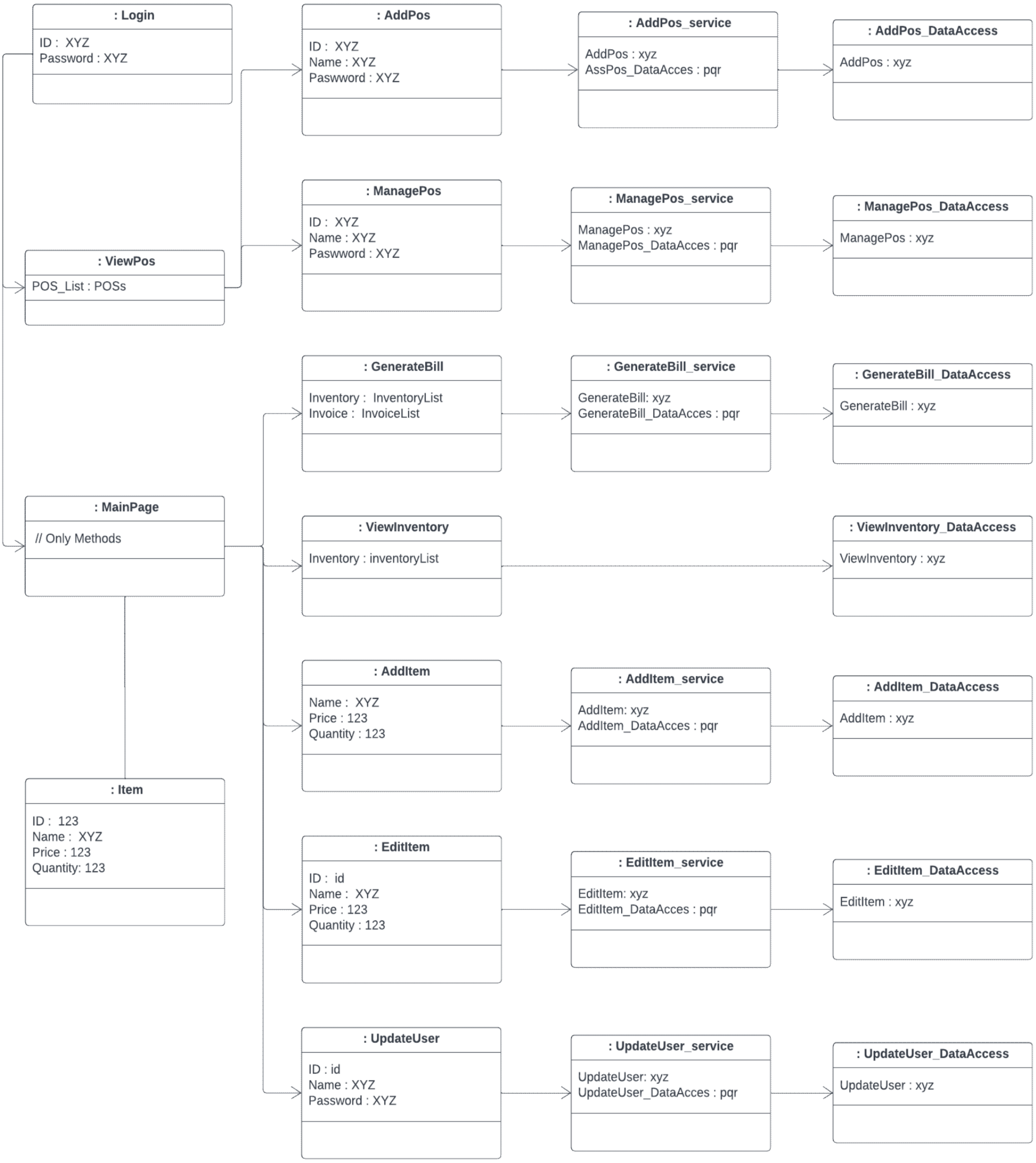
## 1.1.1 Class Diagram



## 1.2 Responsibilities - Usecase Diagram



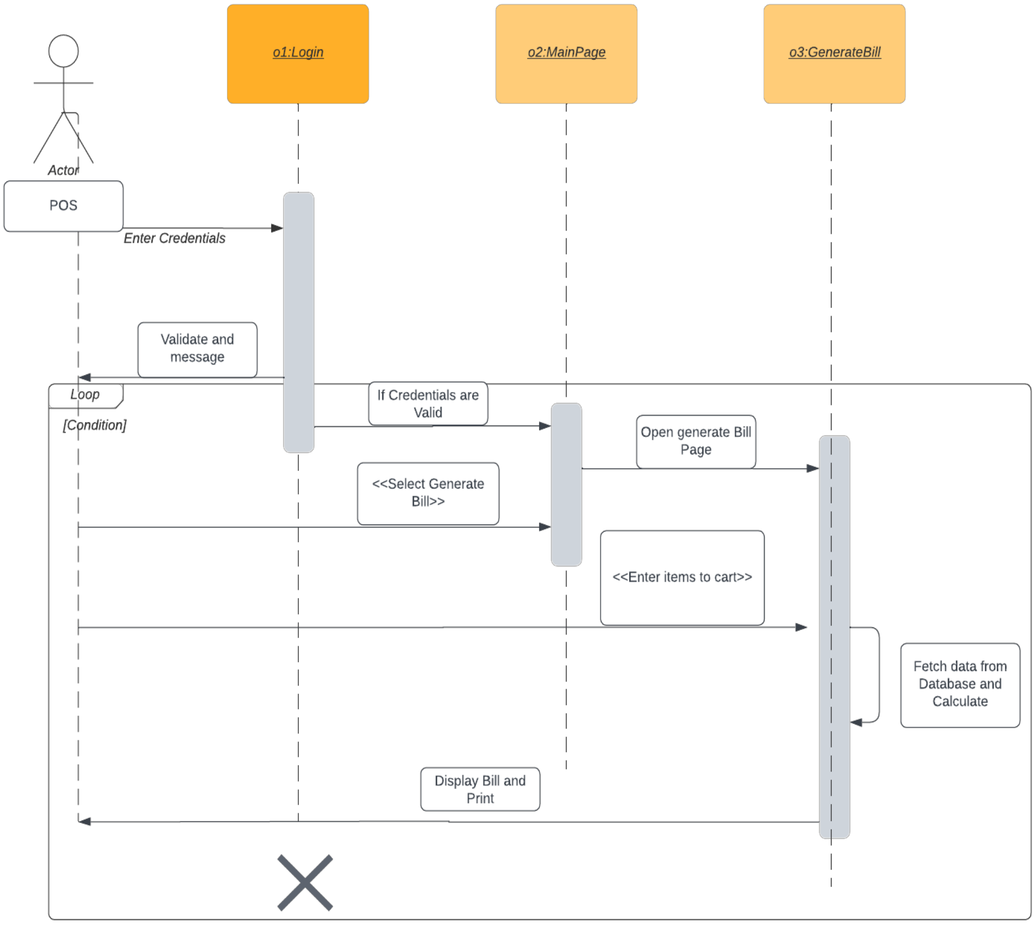
# 1.3 Static snapshot of the system - Object Diagram



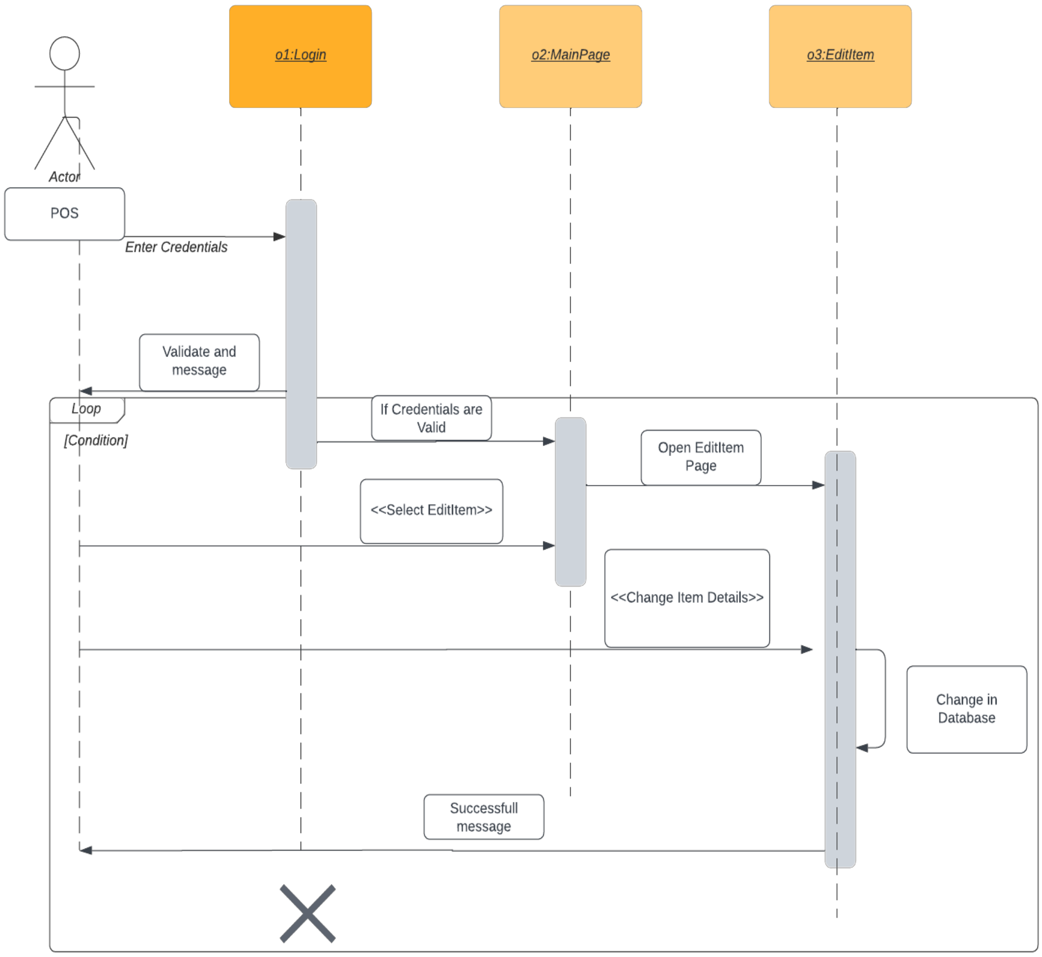
## 1.4 System Interactions through Sequence Diagrams

Sequence diagrams are interaction diagrams that show the sequence of messages exchanged by the set of objects performing a certain task. A sequence diagram shows, as parallel vertical lines (lifeline), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur.

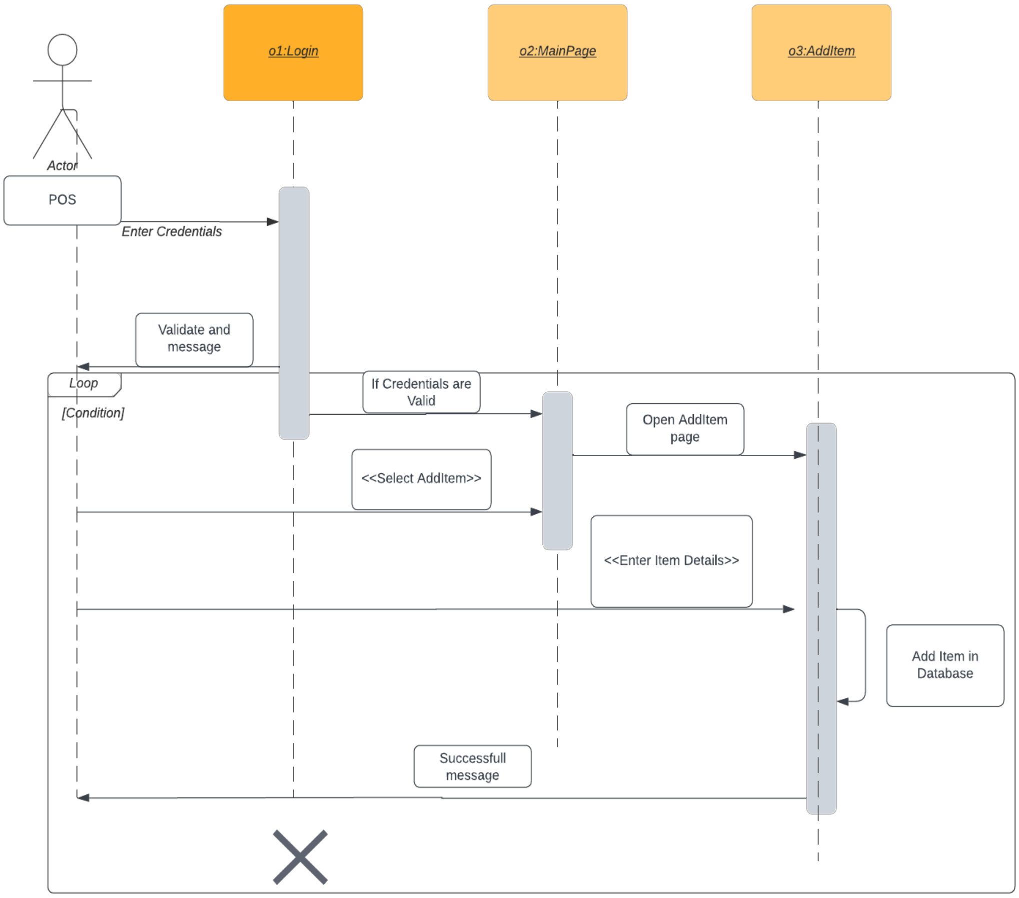
### 1.4.1 **SEQUENCE DIAGRAM #1 - GENERATE-BILL**



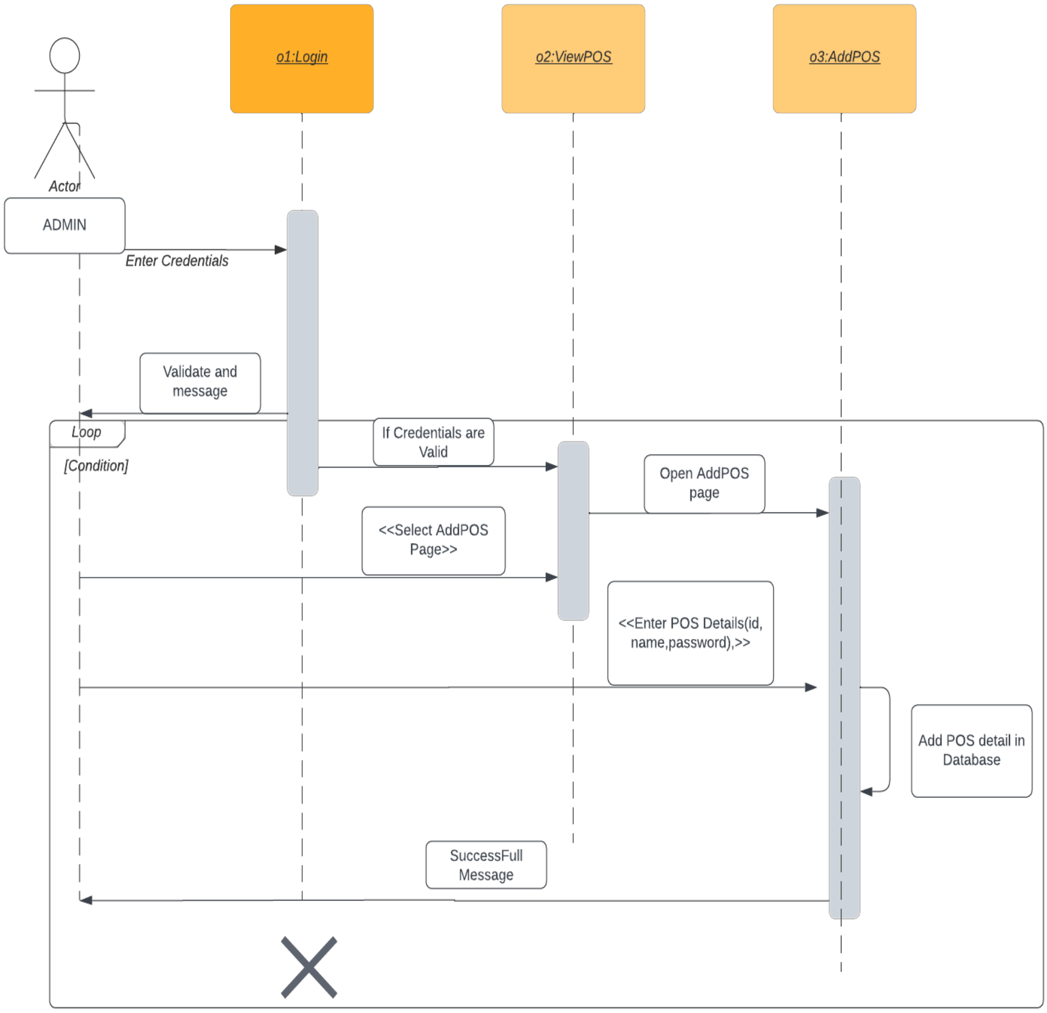
## 1.4.2 **SEQUENCE DIAGRAM #2 - EDIT-ITEMS**



## 1.4.3 **SEQUENCE DIAGRAM #3 - ADD ITEMS**

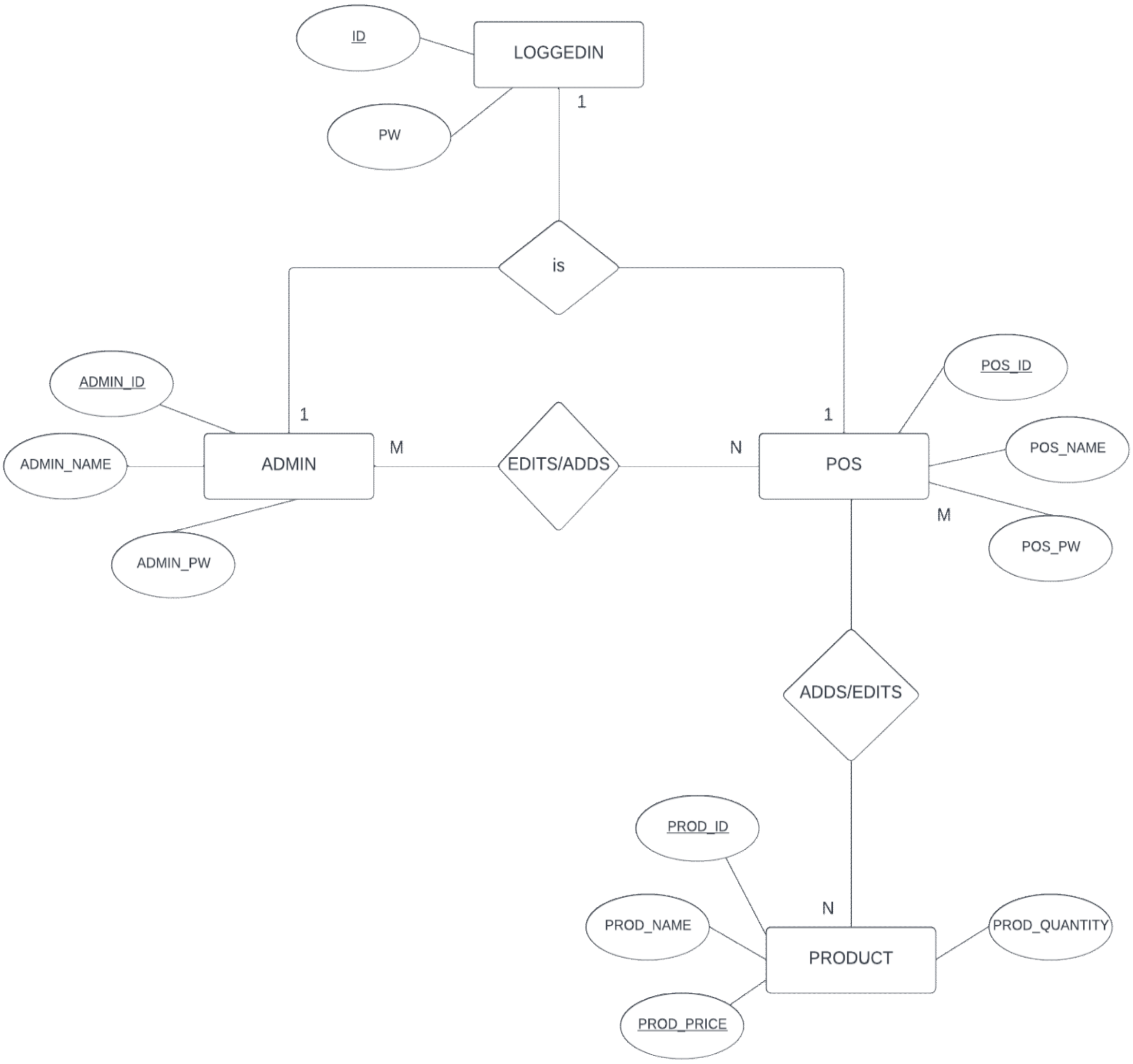


## 1.4.4 **SEQUENCE DIAGRAM #4 - ADD POS**



# 2. Database Design

## 2.1 ER Diagram



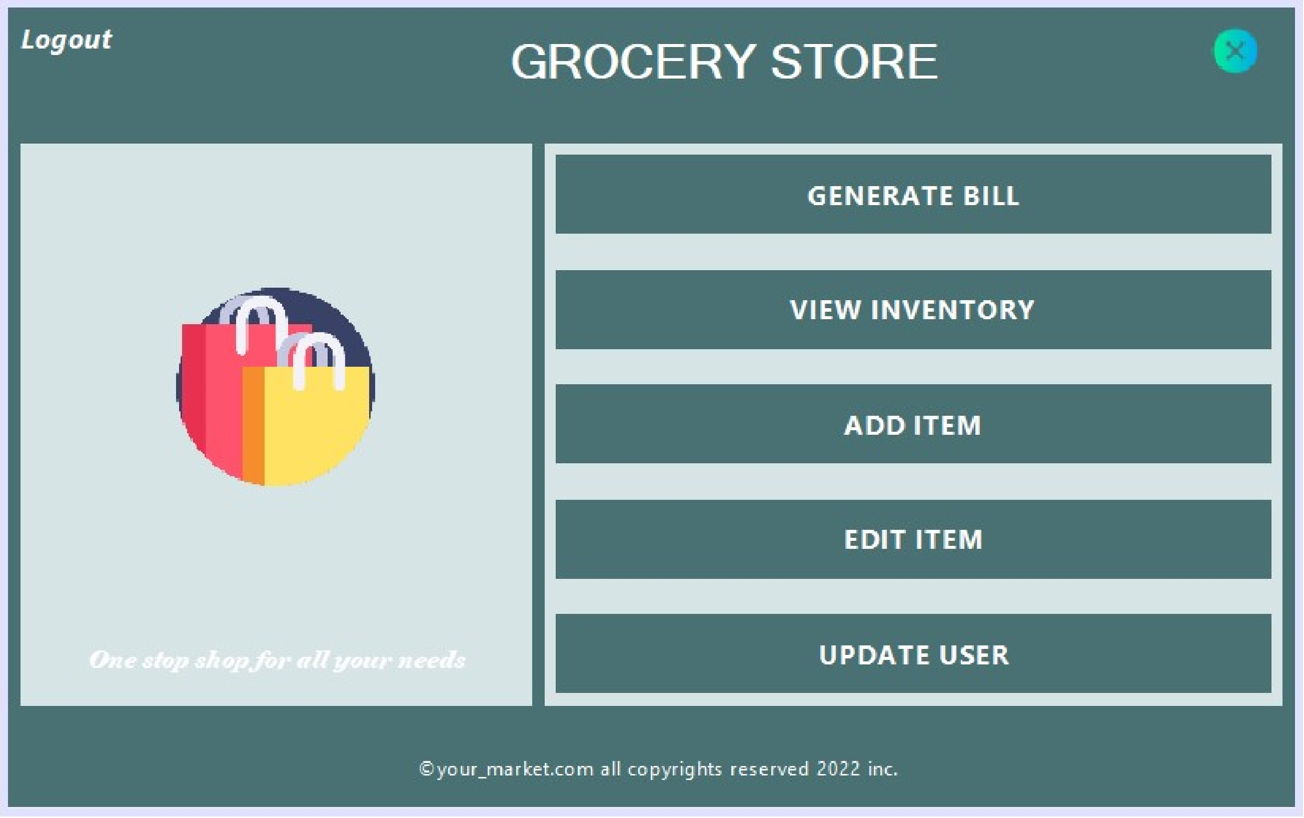
# 3. Implementation Plans

## 3.1 Technology Stack

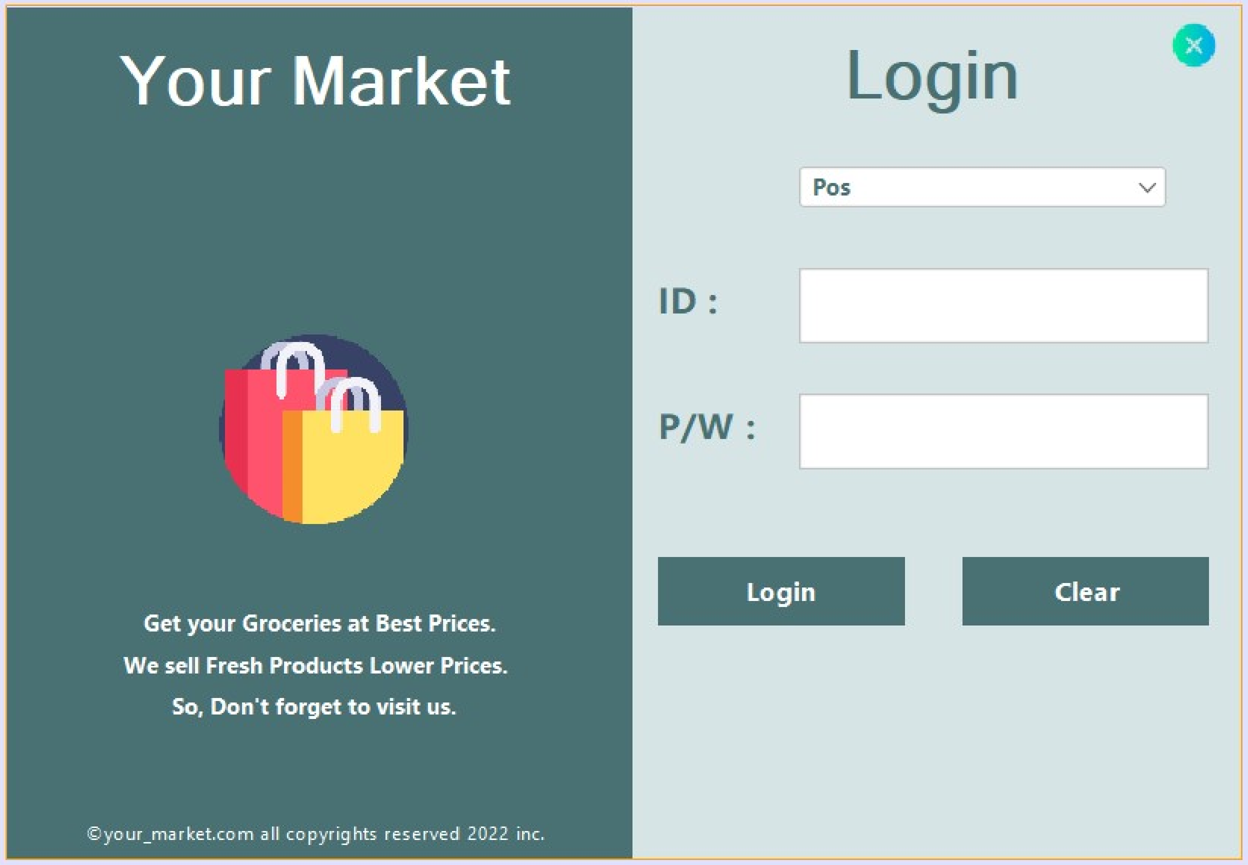
Netbeans IDE has been used for the purpose of developing the application. It uses swing Jforms for UI design and java JDK platform for development. The language it uses is java.
For the purpose of backend development, java is used for development and SQL has been used as Database. SQL queries has been used for the purpose of communication with Database server.

## 3.2User Interface Prototyping

1.Main Page for application.


Logout
GROCERY STORE
GENERATE BILL
VIEW INVENTORY
ADD ITEM
EDIT ITEM
UPDATE USER
One stop shop for all your needs
©your_market.com all copyrights reserved 2022 inc.

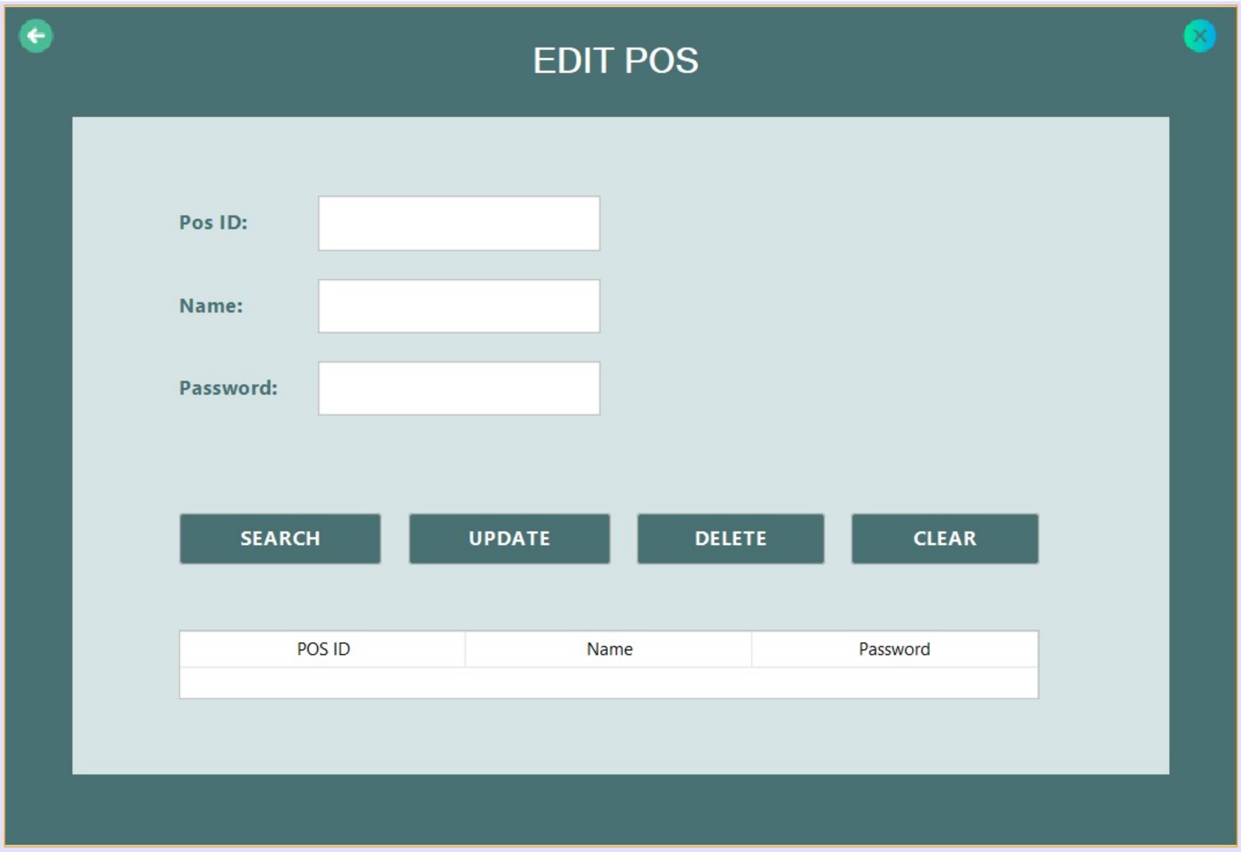
2. Login page UI.



Get your Groceries at Best Prices.
We sell Fresh Products Lower Prices.
So, Don't forget to visit us.

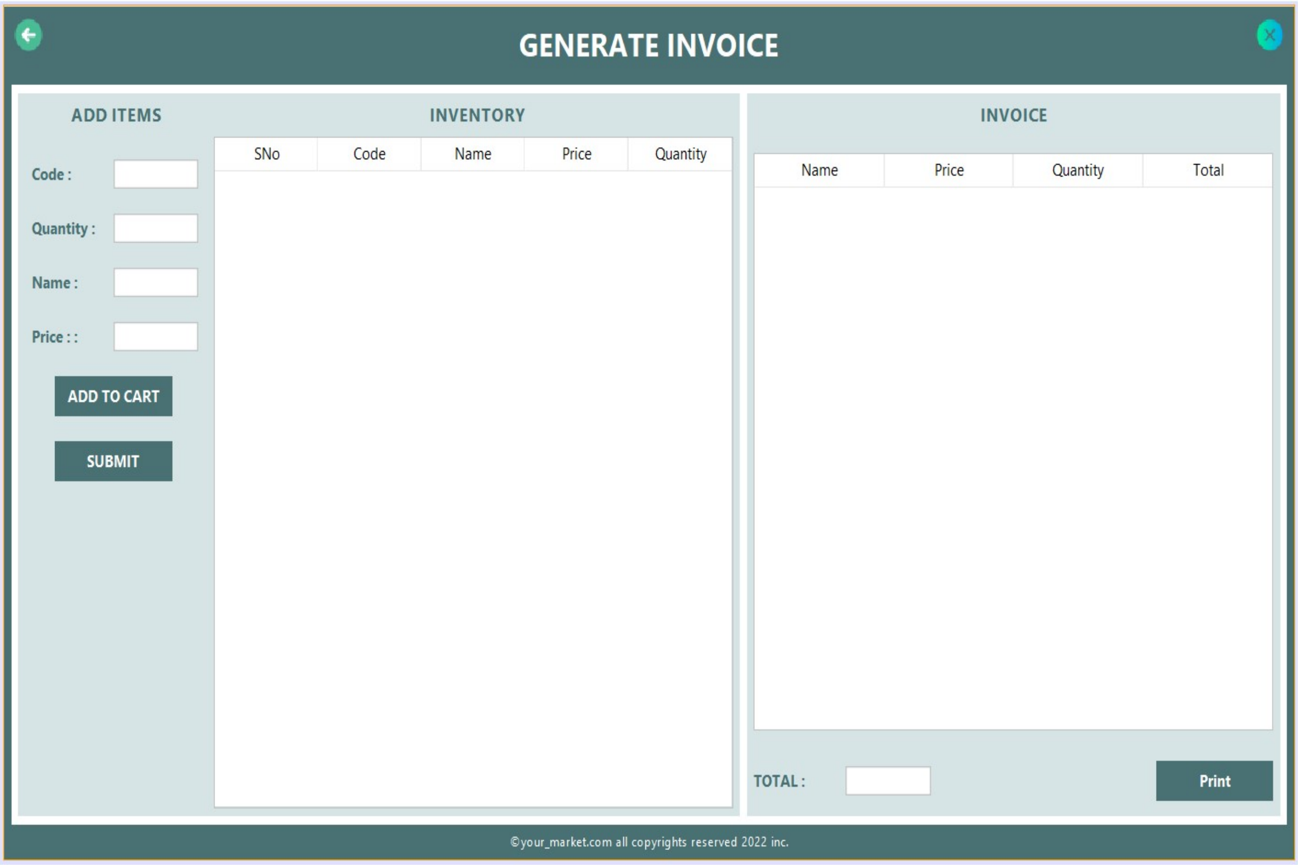
©your_market.com all copyrights reserved 2022 inc.

3. Edit POS UI (only visible when logged in as ADMIN)

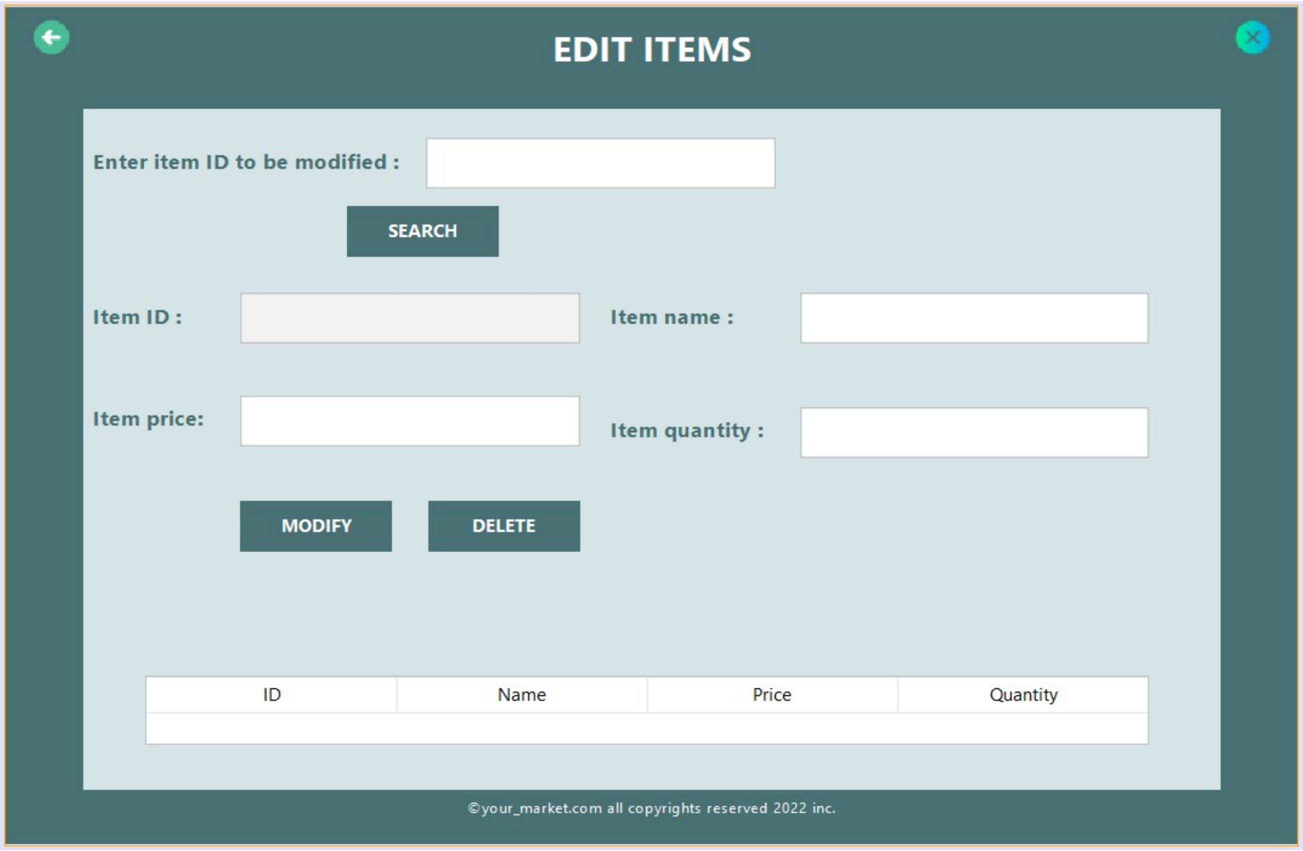


| POS ID | Name | Password |
|---|---|---|
| | | |

4. Generate invoice functionality UI.



5. Edit items menu UI.



# 4. Test Cases

## 4.1 Test Case #1 (TC_LOGIN_01)

**Author: Junaid Ansari**

**Test Case Description:**

Test scenario: Verification of Login
Test case: Enter valid username(ID) and valid password

**Pre-Conditions:**

Need a valid set of Login Credentials, that must be present in the database.

**Test Steps:**

- Select Login type(admin/pos)
- Enter Username
- Enter Password
- Click Login button

**Test Data:**

e.g.,
login-type: admin
Id: admin
Password: x@kjh3

e.g.,
login-type: pos
Id: pos
Password: 2a@34s

**Expected Result:**

**If Login credentials are missing –**

**Output-** (Message)Please Enter the Login details.

**If Login credentials are not in database –**

**Output-** (Message)Please Enter the Valid ID.

**If Login credentials are in database but entered password is wrong–**

**Output-** (Message)Please Enter correct password.

**If Login credentials are found in database –**

**Output-** Main-page appeared when POS is selected as login-type.
**Output-** View-POS appeared when admin is selected as login-type.

**Post Condition:**

Main-page is appeared when POS is logging in.
View-POS is appeared when Admin is logging in.

## 4.2 Test Case #2 (TC_GENERATE_BILL_02)

**Author: Junaid Ansari**

**Test Case Description:**

Test scenario: Generating bill
Test case: Select some items from inventory list with quantity that customer wants to buy.

**Pre-Conditions:**

The item that customer wants to buy, should be present in Inventory(Database).
The logged in user type should be POS.

**Test Steps:**

- Go to Generate Bill
- Choose items from inventory by clicking on table
- Enter Quantity
- Press Add to Cart button
- After selecting all the items, press Submit button.
- Press Print button to download the Bill.

**Test Data:**

e.g.,

| | | |
|---|---|---|
| 1 | Mangoes | 12 |
| 2 | Apples | 40 |
| 3 | XYZ | 50 |

**Expected Result:**

**After clicking Submit button-**

**Output-**Total amount shall be calculated and showed.

**After clicking Print button-**

**Output-**Table containing all the purchased items along with the total shall be printed.

**Post Condition:**

Stock available in the inventory(database) shall be updated.
Bill for the customer shall be printed.

## 4.3 Test Case #3 (TC_ADD_ITEM_03)

**Author: Arham Siddiki**

**Test Case Description:**

Test scenario: Add items to database
Test case: Enter details of items to be added to the database

**Pre-Conditions:**

The logged in user type should be POS.

**Test Steps:**

- Go to Add Items
- Enter the details of item to be added(Name, price pr unit, Stock)
- Press Add button

**Test Data:**

e.g.,

Tomato 10 10

**Expected Result:**

**If item is present in database-**

**Output-** (Message)Item is already in Inventory.

**If item not present in database-**

**Output-** Item shall be added to the inventory(database).

**Post Condition:**

Item Shall be added to the Inventory.

## 4.4 Test Case #4 (TC_EDIT_ITEM_04)

**Author: Arham Siddiki**

**Test Case Description:**

Test scenario: Edit items in inventory

Test case: Enter details of items to be Edited to the database

**Pre-Conditions:**

The logged in user type should be POS.

The item must be present in the Inventory.

**Test Steps:**

- Go to Add Items
- Enter the details of item to be added(Name, price pr unit, Stock)
- Press Add button

**Test Data:**

e.g.,

Tomato 10 10

**Expected Result:**

**If item is present in database-**

**Output-** (Message)Item is already in Inventory.

**If item not present in database-**

**Output-** Item shall be added to the inventory(database).

**Post Condition:**

Item Shall be added to the Inventory.

## 4.5 Test Case #5 (TC_SEARCH_ITEM_05)

**Author: Karan Sachinkumar**

**Test Case Description:**

Test scenario: Search items in inventory.
Test case: Enter ID of items to be searched in the inventory.

**Pre-Conditions:**

The logged in user type should be POS.
The item must be present in the Inventory.

**Test Steps:**

- Go to Edit Items
- Enter the id of the item to be searched
- Click search button

**Test Data:**

e.g.
2

**Expected Result:**

**If item not present in inventory-**

**Output-** (Message)Item not present in Inventory.

**If item is present in inventory(database)-**

**Output-** Item shall be appeared in table.

**Post Condition:**

Item Shall be appeared in the Inventory.

## 4.6 Test Case #6 (TC_UPDATE_USER_06)

**Author: Karan Sachinkumar**

**Test Case Description:**

Test scenario: Updating User
Test case: Enter the details of user.

**Pre-Conditions:**

The logged in user type should be POS.

**Test Steps:**

- Go to Update User
- Enter the details of user(ID, Name, password)
- Press Update button

**Test Data:**

e.g.

| | | |
|---|---|---|
| User1 | Sachin | 1c$12@ |

**Expected Result:**

**If entered ID is different from user's ID-**

**Output-** (Message)You can update yourself only.

**If ID is matched with user's ID-**

**Output-** User shall be updated.
(Message)Updated successfully.

**Post Condition:**

User's details should be updated in database.

# References

//No references are here for now. Once used, will be updated here.//