08/06/2022

PHARMACEUTICAL INVENTORY MANAGEMENT SYSTEM REPORT

INTE 12223 - Database Design and Development (2019/2020)

INTE 12213 - Object Oriented Programming (2019/2020)

Prepared By

Group 02

Table of Contents;

Introduction

This project is focused on the automation of a pharmaceutical system. In this system, we can handle the following tasks;

- Maintaining records(stock) of medicines.
- Handle user(employee) details.
- Control the access to the database of employee data.
- Tracking the drugs in storage
- Check the validity (expired batches) of drugs
- Issuing the short expiry drugs earlier
- Calculate the day-end sales.
- Calculate the cash in hand.
- Trace the outflow of cash from the business.

The main aim of the project is the automation of the database of the pharmaceutical shop. This project is an insight into the design and implementation of a Pharmacy Management System. This has been done by creating a database of the available medicines and the users in the shop. The primary aim of the pharmacy management system is to improve accuracy as well as enhance the safety and efficiency in the pharmaceutical store. We have developed this software for ensuring the effective issue of drugs by maintaining the details of the drugs in stock.

Related / similar systems

- FastForward II(Currently used in Abans PLC.)
- McKesson Operational Efficiency for Pharmacies.
- PrimeRx.
- Cerner Retail Pharmacy.

This system is based on the above-mentioned FastForward II system. This has been customized concerning a pharmacy giving some unique features to the system according to the needs of a pharmacy.

Analysis - of the system requirements and needs

Needs of the pharmacy

It was analyzed that a pharmacy has a variety of needs. Among these needs, this system tackles the above-mentioned needs with great lenience. Apart from those, as future developments, the rest of the requirements of the pharmacy will be addressed. Those needs will be

Generating of reports

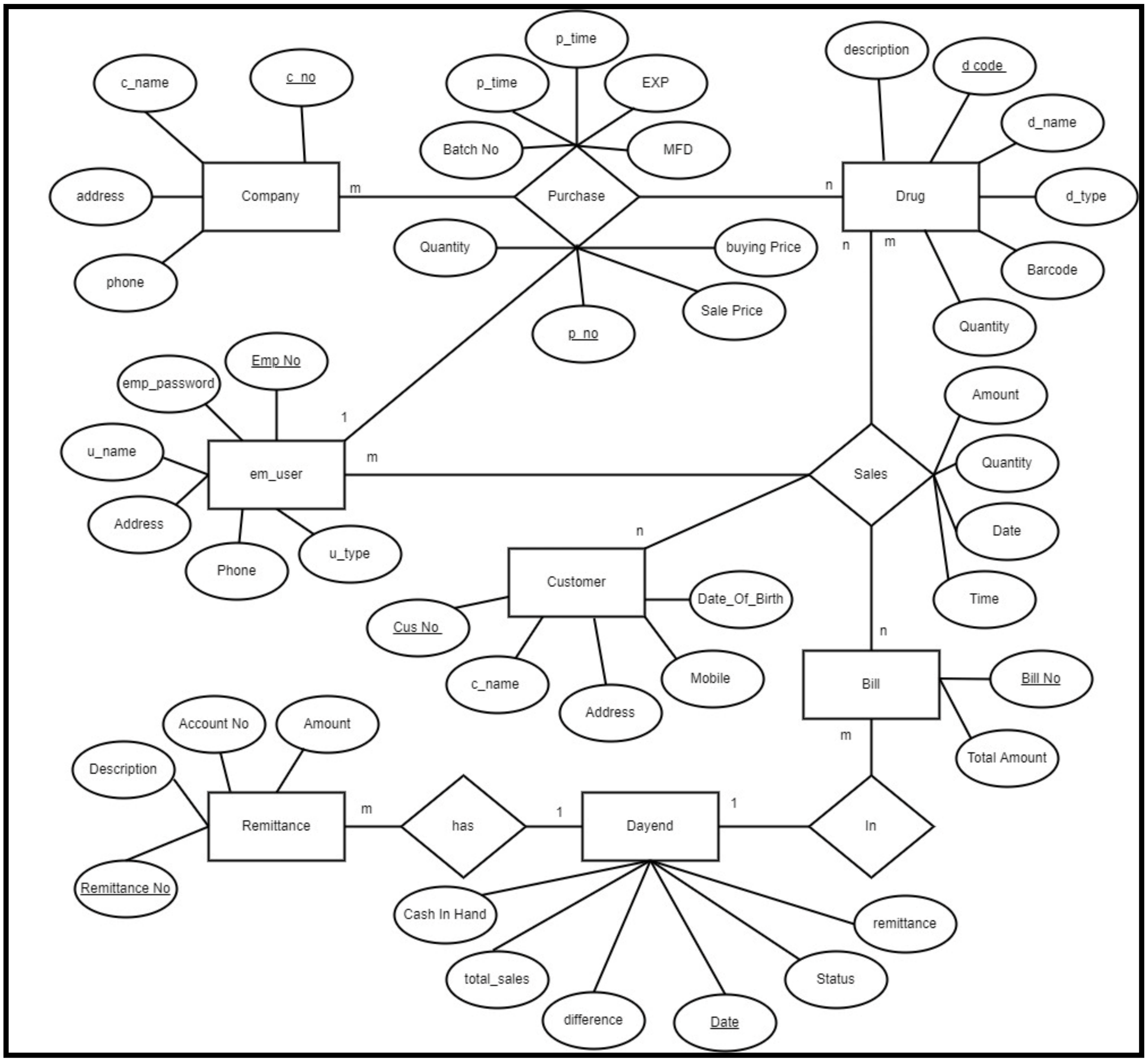
Improve the access control

Cancellation of sales and purchases.

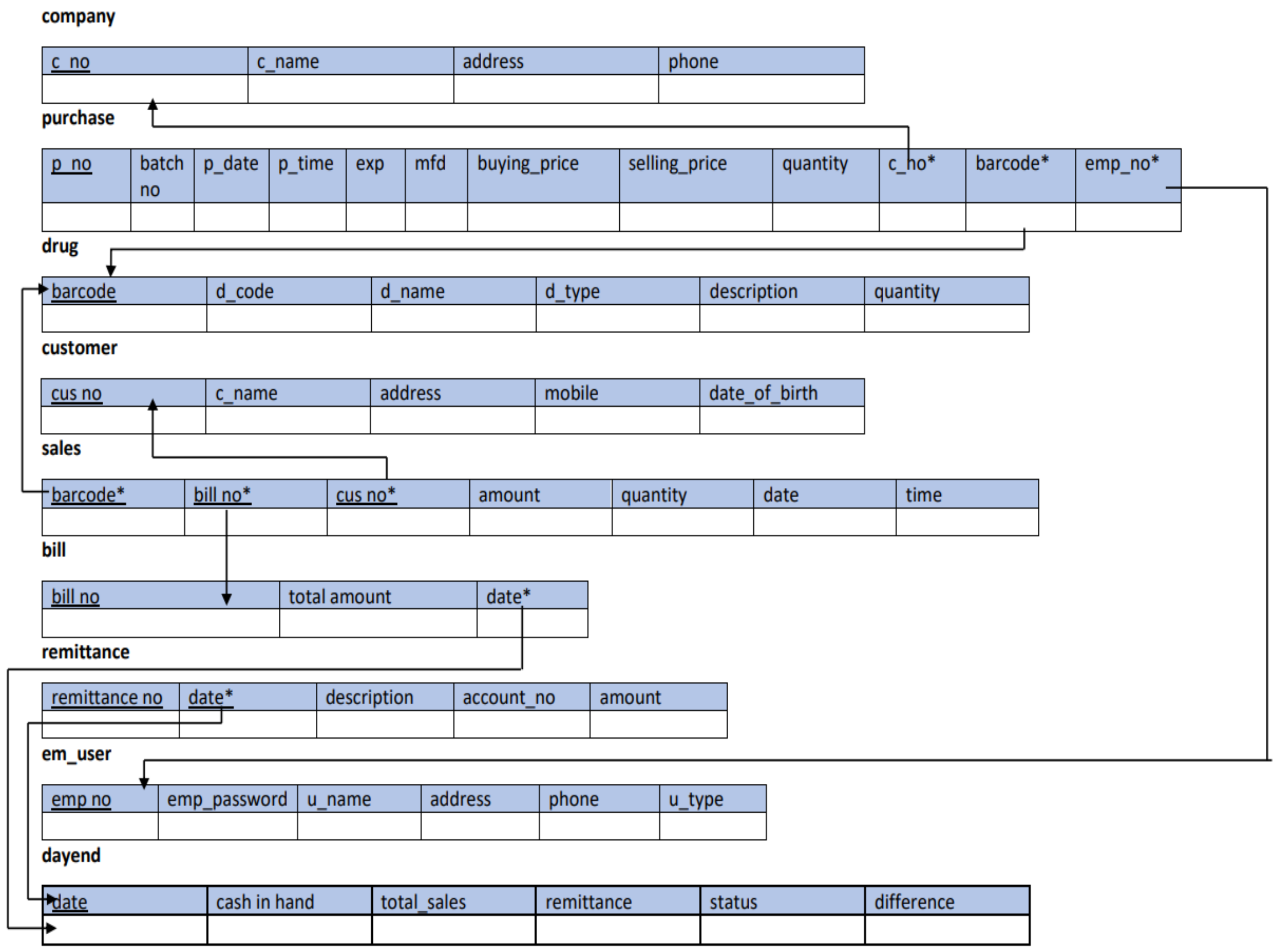
Requirements of the system

- OPERATING SYSTEM: Windows 8 and above
- PROCESSOR: Intel Core 2 Quad CPU Q6600 @ 2.40GHz (4 CPUs) and above
- MEMORY: minimum 2GB
- HDD Space: 1GB(This must be increased depending on the size of the database

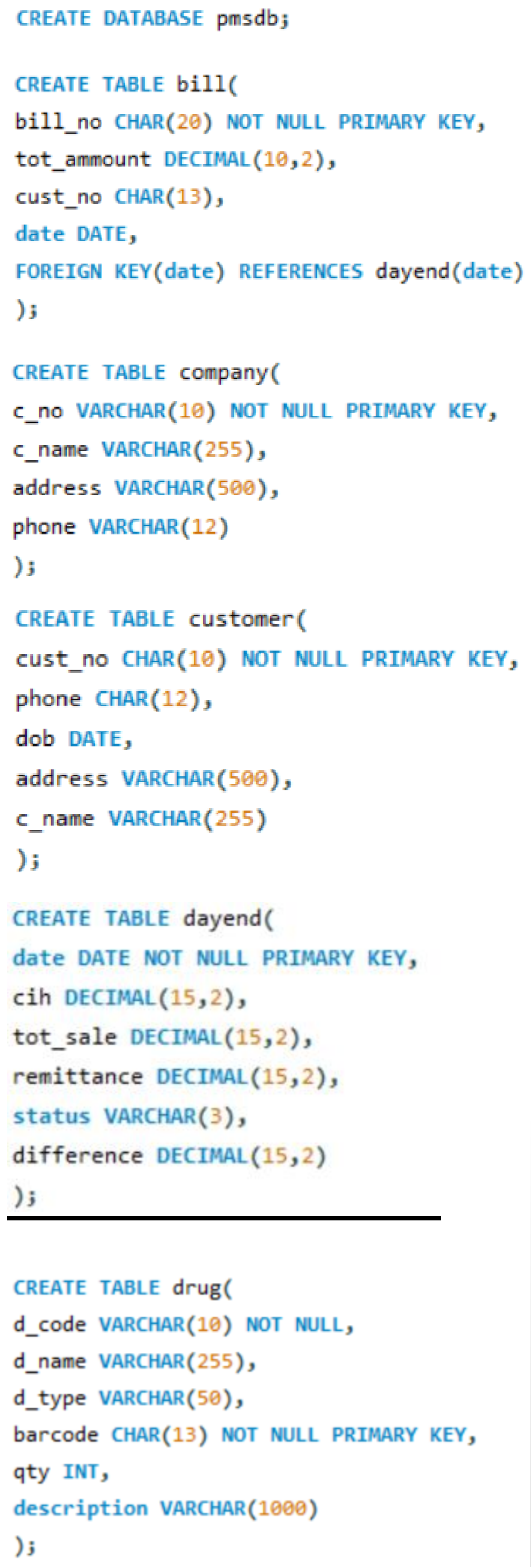
Database design and implementation (database, interface)



Relational Database Schema

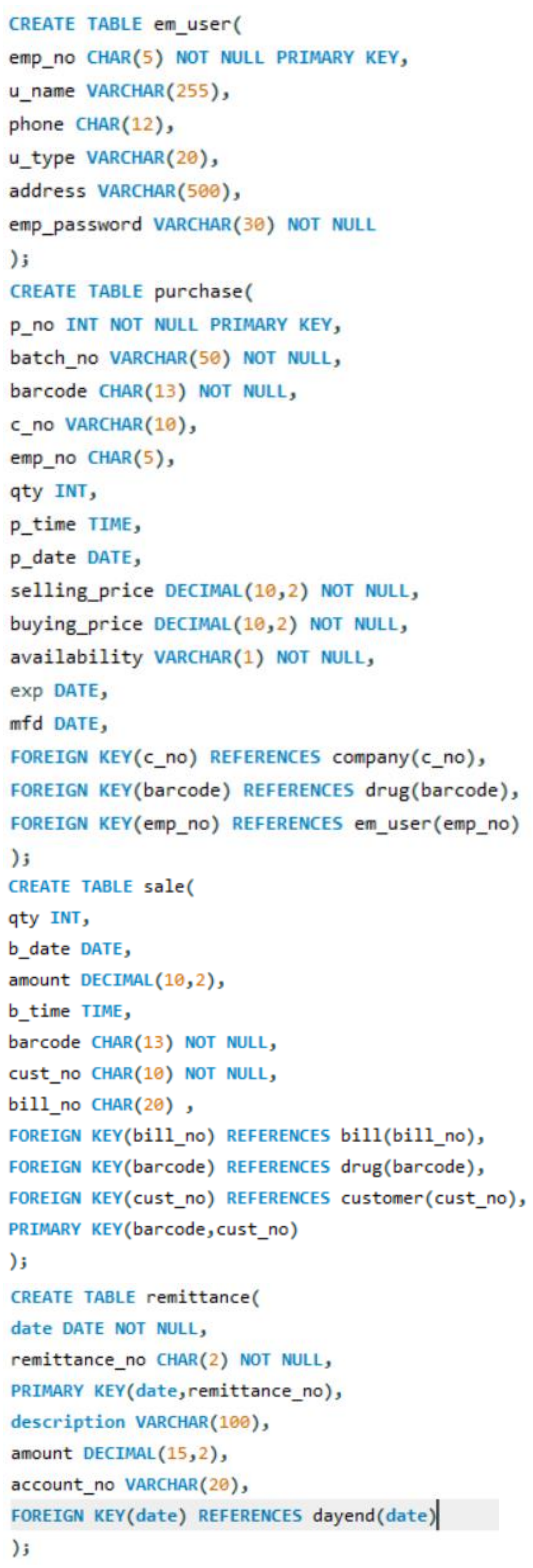


SQL Queries;



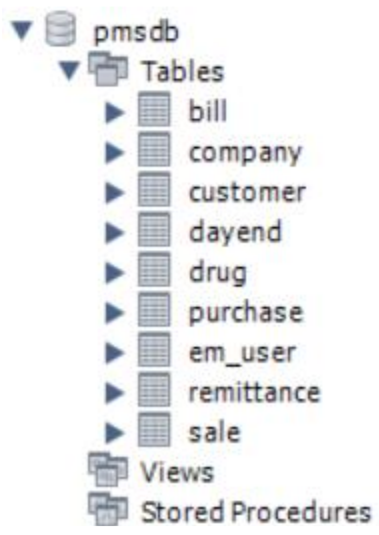
```
CREATE DATABASE pmsdb;

CREATE TABLE bill(
bill_no CHAR(20) NOT NULL PRIMARY KEY,
tot_ammount DECIMAL(10,2),
cust_no CHAR(13),
date DATE,
FOREIGN KEY(date) REFERENCES dayend(date)
);

CREATE TABLE company(
c_no VARCHAR(10) NOT NULL PRIMARY KEY,
c_name VARCHAR(255),
address VARCHAR(500),
phone VARCHAR(12)
);

CREATE TABLE customer(
cust_no CHAR(10) NOT NULL PRIMARY KEY,
phone CHAR(12),
dob DATE,
address VARCHAR(500),
c_name VARCHAR(255)
);

CREATE TABLE dayend(
date DATE NOT NULL PRIMARY KEY,
cih DECIMAL(15,2),
tot_sale DECIMAL(15,2),
remittance DECIMAL(15,2),
status VARCHAR(3),
difference DECIMAL(15,2)
);
```

```
CREATE TABLE drug(
d_code VARCHAR(10) NOT NULL,
d_name VARCHAR(255),
d_type VARCHAR(50),
barcode CHAR(13) NOT NULL PRIMARY KEY,
qty INT,
description VARCHAR(1000)
);
```



```
CREATE TABLE em_user(
emp_no CHAR(5) NOT NULL PRIMARY KEY,
u_name VARCHAR(255),
phone CHAR(12),
u_type VARCHAR(20),
address VARCHAR(500),
emp_password VARCHAR(30) NOT NULL
);
CREATE TABLE purchase(
p_no INT NOT NULL PRIMARY KEY,
batch_no VARCHAR(50) NOT NULL,
barcode CHAR(13) NOT NULL,
c_no VARCHAR(10),
emp_no CHAR(5),
qty INT,
p_time TIME,
p_date DATE,
selling_price DECIMAL(10,2) NOT NULL,
buying_price DECIMAL(10,2) NOT NULL,
availability VARCHAR(1) NOT NULL,
exp DATE,
mfd DATE,
FOREIGN KEY(c_no) REFERENCES company(c_no),
FOREIGN KEY(barcode) REFERENCES drug(barcode),
FOREIGN KEY(emp_no) REFERENCES em_user(emp_no)
);
CREATE TABLE sale(
qty INT,
b_date DATE,
amount DECIMAL(10,2),
b_time TIME,
barcode CHAR(13) NOT NULL,
cust_no CHAR(10) NOT NULL,
bill_no CHAR(20) ,
FOREIGN KEY(bill_no) REFERENCES bill(bill_no),
FOREIGN KEY(barcode) REFERENCES drug(barcode),
FOREIGN KEY(cust_no) REFERENCES customer(cust_no),
PRIMARY KEY(barcode,cust_no)
);
CREATE TABLE remittance(
date DATE NOT NULL,
remittance_no CHAR(2) NOT NULL,
PRIMARY KEY(date,remittance_no),
description VARCHAR(100),
amount DECIMAL(15,2),
account_no VARCHAR(20),
FOREIGN KEY(date) REFERENCES dayend(date)
);
```

Final Database;


```
▼ pmsdb
    ▼ Tables
        ▶ bill
        ▶ company
        ▶ customer
        ▶ dayend
        ▶ drug
        ▶ purchase
        ▶ em_user
        ▶ remittance
        ▶ sale
      Views
      Stored Procedures
```

<u>System implementation</u>

This system has been implemented based on OOP concepts and with the help of the JavaFX application. JavaFX was used to create the interfaces and the backend for the system.

- Maintaining records(stock) of medicines.
    - The system identifies different types of medicines from each purchase into the system and stores that information in a drug table .

- Handle user(employee) details.
    - The employee details are stored in the database in a separate table and this table can be edited depending on the employee (new data can be added or the existing data can be changed)

- Control the access to the database of employee password.
    - Editing the access password to the system can only done by the employee himself/herself.

- Tracking the drugs in storage
    - Using the inventory tracker function the users of the software can check the available number of stocks within the pharmacy.

- Check the validity (expired batches) of drugs
    - This also can be checked using the tracker function.

- Issuing the short expiry drugs earlier
    - Drugs are issued using FIFO (First in first out) method to avoid short expiry drugs being stocked in the system. But in this system, if a drug with short-term expiry was added to the system later it will not be issued earlier. This will be resolved as future development.

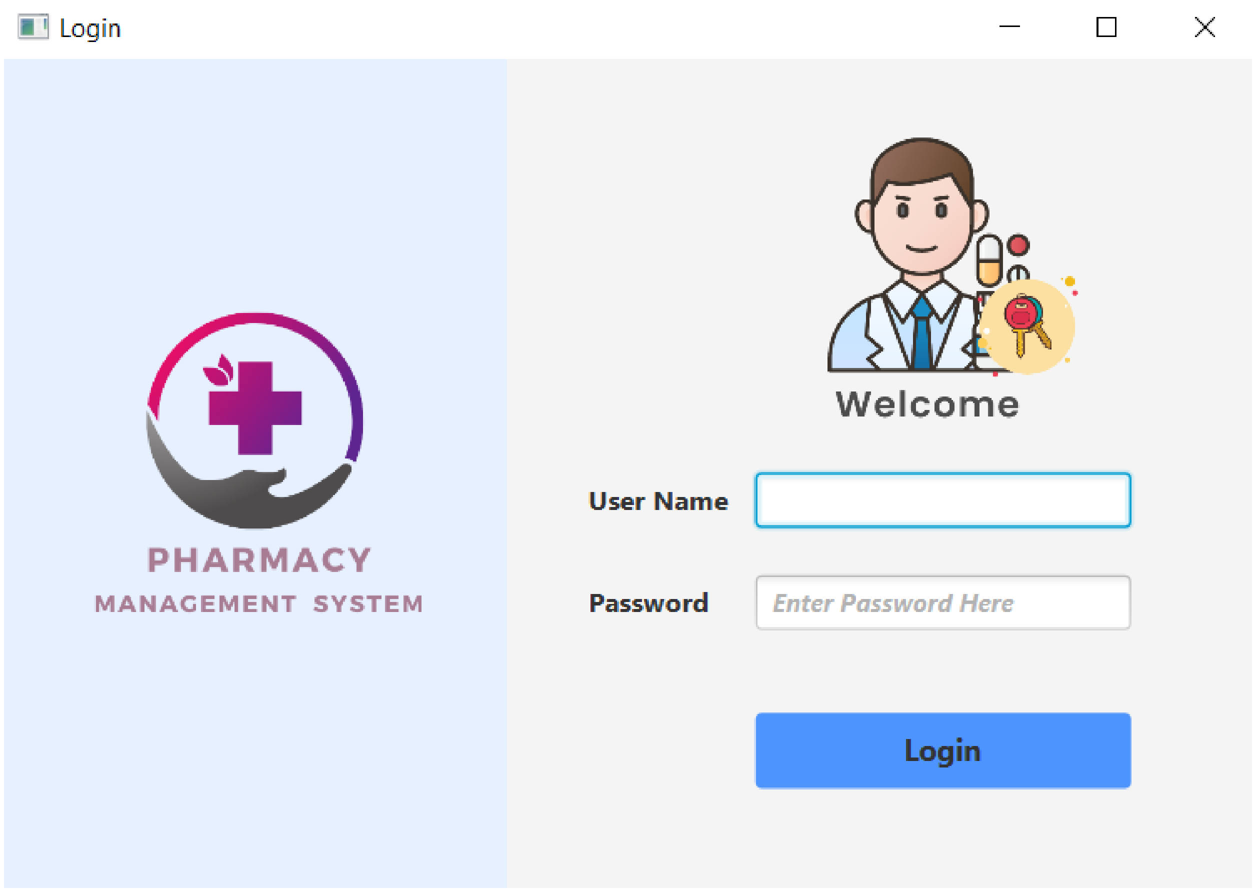
- Calculate the day-end sales.
    - Using the sales for the day the system calculates the total sales at the end of the day.

- Trace the outflow of cash from the business.
    - When the user inputs the outflows of cash as remittance, the system records them and by the day’s end all these can be displayed.

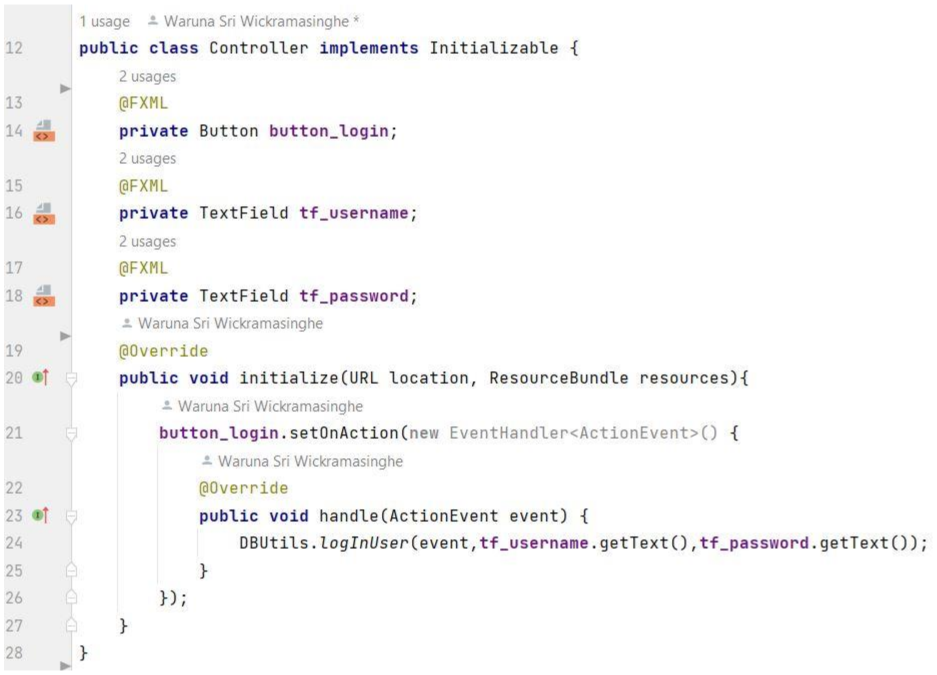
- Calculate the difference of cash in hand.
    - After the user has inputted the cash outflow(remittance) from the business the system itself calculates the cash that should be retained in the business as the difference in that window. After the user gives the real amount of cash in hand as an input the difference amount will be changed. With this functionality, the user can check whether the sales and the cash in hand are tallies with the outflows of cash from the pharmacy.

System Functionality and User Interfaces

The first interface the user of the software interacts with is the following dialogue box.

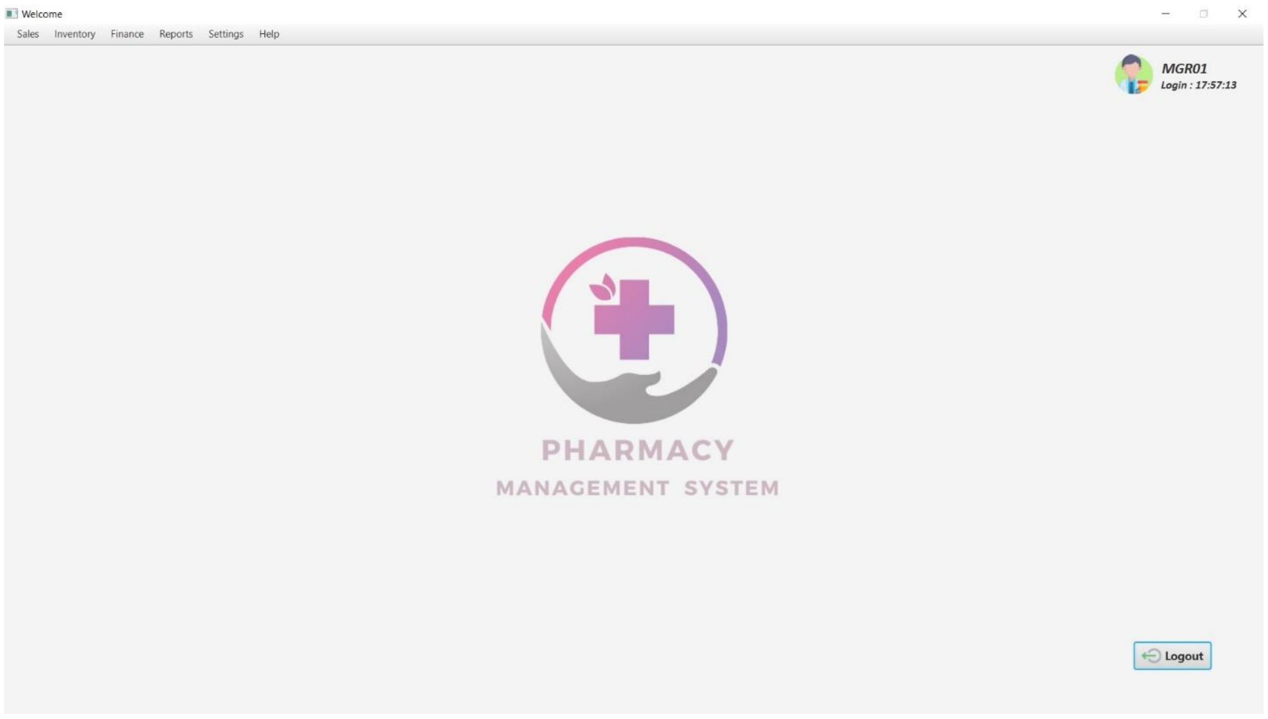


This dialogue box enables the user to input his credentials and log into the system. The above FXML interface interacts with the below controller class once the user presses the login button. The controller class will detect the action event and call the “loginuser” method in DButils class.

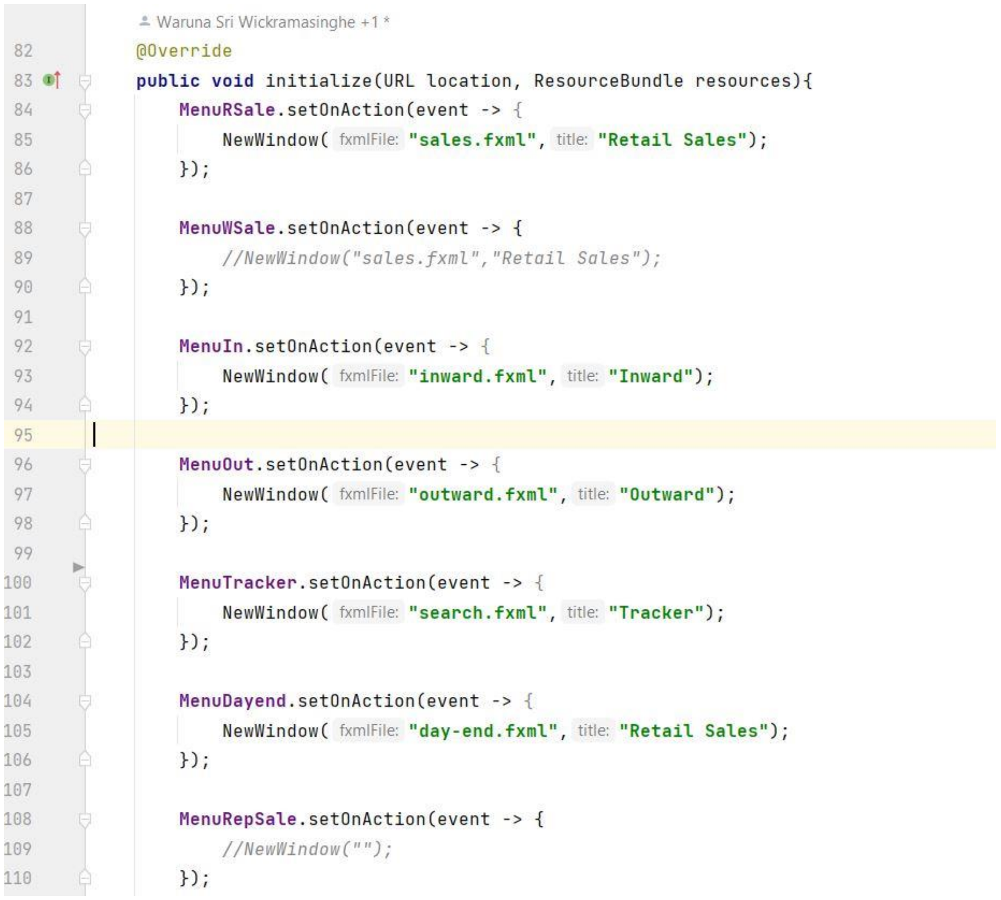


```
1 usage   Waruna Sri Wickramasinghe *
public class Controller implements Initializable {
    2 usages
    @FXML
    private Button button_login;
    2 usages
    @FXML
    private TextField tf_username;
    2 usages
    @FXML
    private TextField tf_password;
    Waruna Sri Wickramasinghe
    @Override
    public void initialize(URL location, ResourceBundle resources){
        Waruna Sri Wickramasinghe
        button_login.setOnAction(new EventHandler<ActionEvent>() {
            Waruna Sri Wickramasinghe
            @Override
            public void handle(ActionEvent event) {
                DBUtils.logInUser(event,tf_username.getText(),tf_password.getText());
            }
        });
    }
}
```

This loginuser method is used to retrieve username and password from the database in relation to the entered username. Then those are compared and the system moves forward (When the system is initially implemented the developing team will release a predefine username and a password for the system) , then the user is moved on to the welcome screen in which he/she can choose the functionality of the system which he expects to perform. In simpler terms this screen acts as the navigator to the system

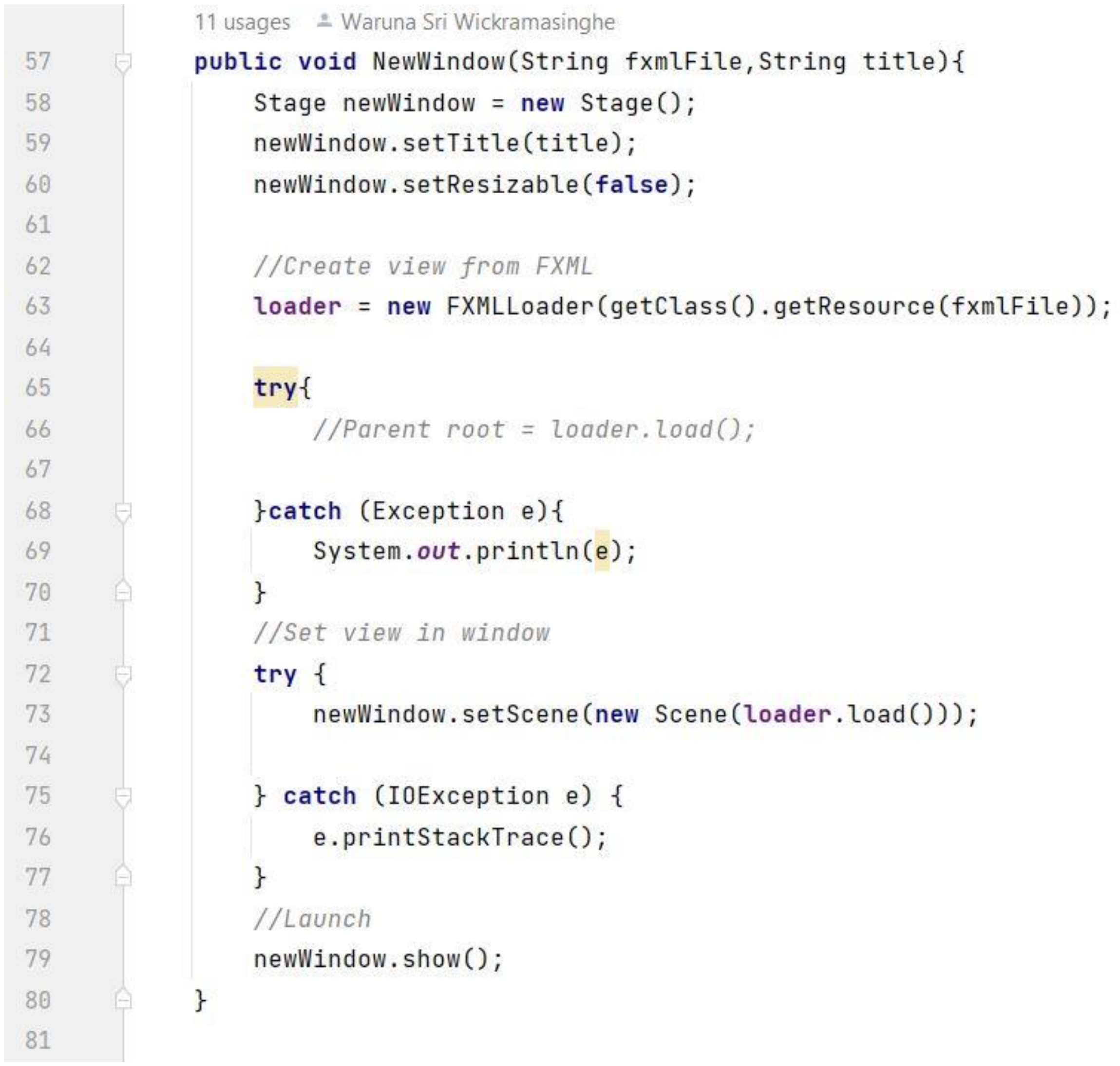


The following is the controller class related to the above FXML. This class detects the action events related to the menu bar menu items. According to the user action using the “NewWindow” method in “welcome_controller” class, the system opens new FXML interfaces.

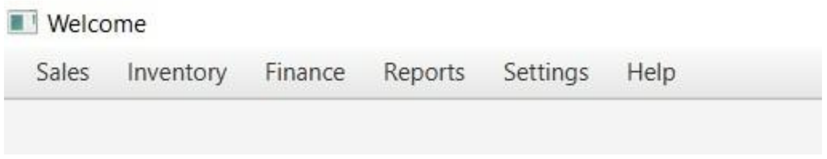


```
Waruna Sri Wickramasinghe +1 *
@Override
public void initialize(URL location, ResourceBundle resources){
    MenuRSale.setOnAction(event -> {
        NewWindow( fxmlFile: "sales.fxml", title: "Retail Sales");
    });

    MenuWSale.setOnAction(event -> {
        //NewWindow("sales.fxml","Retail Sales");
    });

    MenuIn.setOnAction(event -> {
        NewWindow( fxmlFile: "inward.fxml", title: "Inward");
    });

    MenuOut.setOnAction(event -> {
        NewWindow( fxmlFile: "outward.fxml", title: "Outward");
    });

    MenuTracker.setOnAction(event -> {
        NewWindow( fxmlFile: "search.fxml", title: "Tracker");
    });

    MenuDayend.setOnAction(event -> {
        NewWindow( fxmlFile: "day-end.fxml", title: "Retail Sales");
    });

    MenuRepSale.setOnAction(event -> {
        //NewWindow("");
    });
```

The “Newwindow” method is as follows:

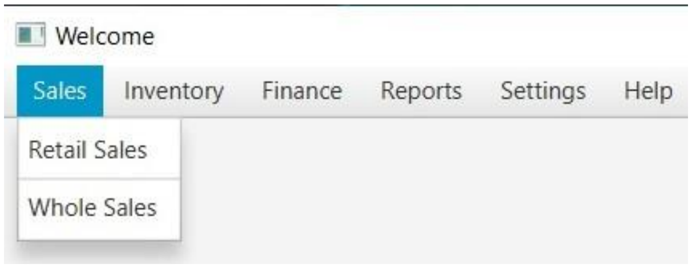

```
11 usages   Waruna Sri Wickramasinghe
public void NewWindow(String fxmlFile,String title){
    Stage newWindow = new Stage();
    newWindow.setTitle(title);
    newWindow.setResizable(false);

    //Create view from FXML
    loader = new FXMLLoader(getClass().getResource(fxmlFile));

    try{
        //Parent root = loader.load();

    }catch (Exception e){
        System.out.println(e);
    }
    //Set view in window
    try {
        newWindow.setScene(new Scene(loader.load()));

    } catch (IOException e) {
        e.printStackTrace();
    }
    //Launch
    newWindow.show();
}
```

The system has the following tools (functionalities).

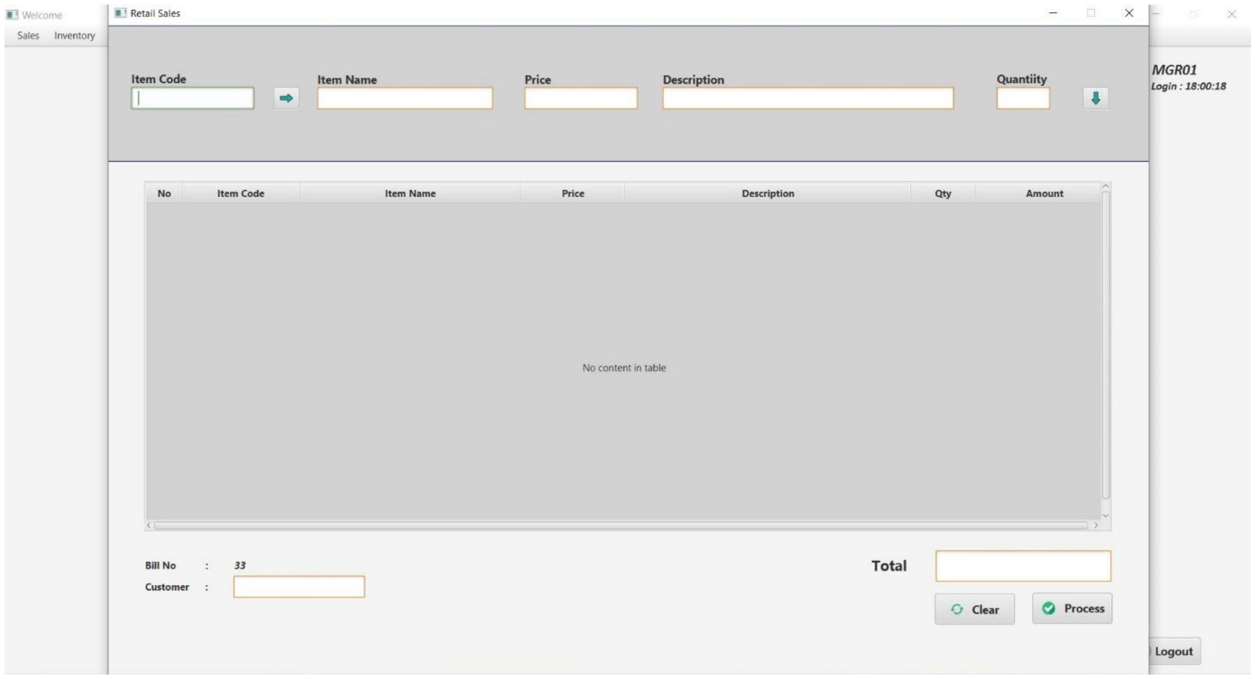


The user can choose each main function and he/she will have the ability to access the subfunctions within that main menu icon.

As mentioned above if the user chooses “Sales” function, he/she will be directed to the following toolbar



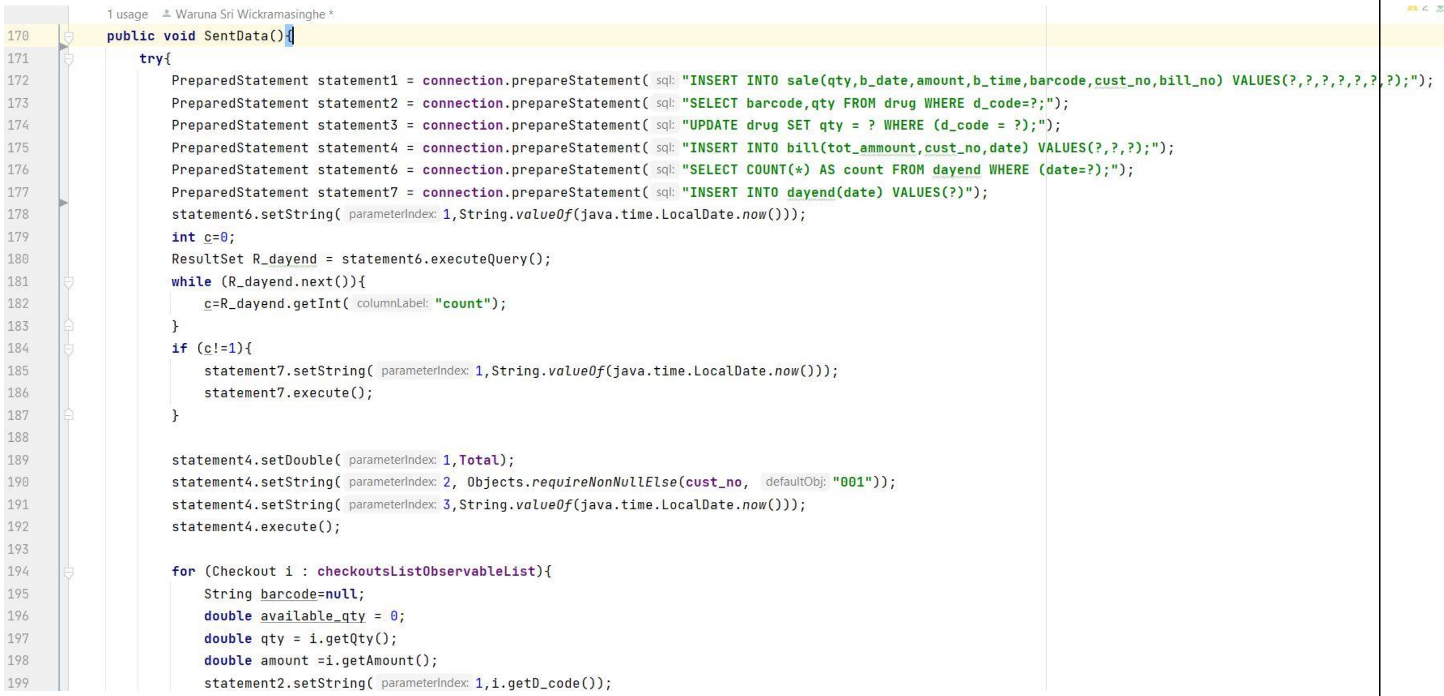
The wholesale function will be developed in the next stage of the system development. Therefore, the user currently only could perform retails sales. And the following FXML will be opened.



Then user should type in the Item code field the item code relevant to the drug he/she is issuing when he/she clicks on the green arrow the Item Name ,Price and Description fields will be automatically filled. Then the user can insert the quantity (**This quantity must be less than the available amount and if not will give an error message**).After that he/she can add that entry in to the sale and after repeating the process for any amount of times needed the sale can be finalized using the process button.

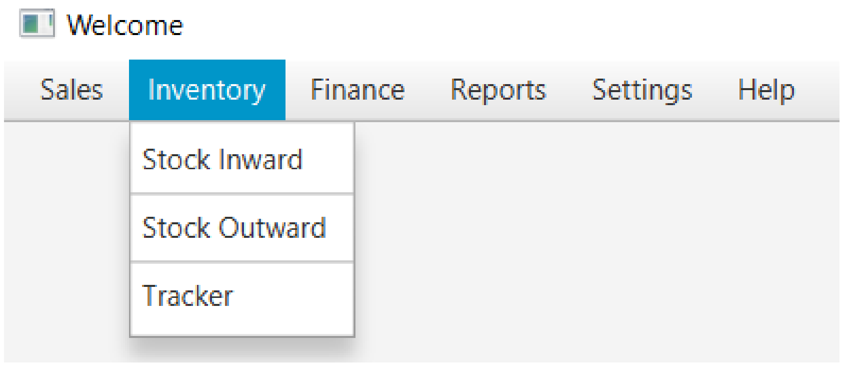
After the process button is pressed the database is manipulated according to the sale made using the “SentData” method in “welcome_controller” class. This updates the database as follows:

- Updating the quantity in drug table
- Insert data into sale and bill tables.



```
1 usage   Waruna Sri Wickramasinghe *
public void SentData(){
    try{
        PreparedStatement statement1 = connection.prepareStatement( sql: "INSERT INTO sale(qty,b_date,amount,b_time,barcode,cust_no,bill_no) VALUES(?,?,?,?,?,?,?);");
        PreparedStatement statement2 = connection.prepareStatement( sql: "SELECT barcode,qty FROM drug WHERE d_code=?;");
        PreparedStatement statement3 = connection.prepareStatement( sql: "UPDATE drug SET qty = ? WHERE (d_code = ?);");
        PreparedStatement statement4 = connection.prepareStatement( sql: "INSERT INTO bill(tot_ammount,cust_no,date) VALUES(?,?,?);");
        PreparedStatement statement6 = connection.prepareStatement( sql: "SELECT COUNT(*) AS count FROM dayend WHERE (date=?);");
        PreparedStatement statement7 = connection.prepareStatement( sql: "INSERT INTO dayend(date) VALUES(?)");
        statement6.setString( parameterIndex: 1,String.valueOf(java.time.LocalDate.now()));
        int c=0;
        ResultSet R_dayend = statement6.executeQuery();
        while (R_dayend.next()){
            c=R_dayend.getInt( columnLabel: "count");
        }
        if (c!=1){
            statement7.setString( parameterIndex: 1,String.valueOf(java.time.LocalDate.now()));
            statement7.execute();
        }

        statement4.setDouble( parameterIndex: 1,Total);
        statement4.setString( parameterIndex: 2, Objects.requireNonNullElse(cust_no,  defaultObj: "001"));
        statement4.setString( parameterIndex: 3,String.valueOf(java.time.LocalDate.now()));
        statement4.execute();

        for (Checkout i : checkoutsListObservableList){
            String barcode=null;
            double available_qty = 0;
            double qty = i.getQty();
            double amount =i.getAmount();
            statement2.setString( parameterIndex: 1,i.getD_code());
```

If the user chooses “Inventory” function, he/she will be directed to the following menu bar ;

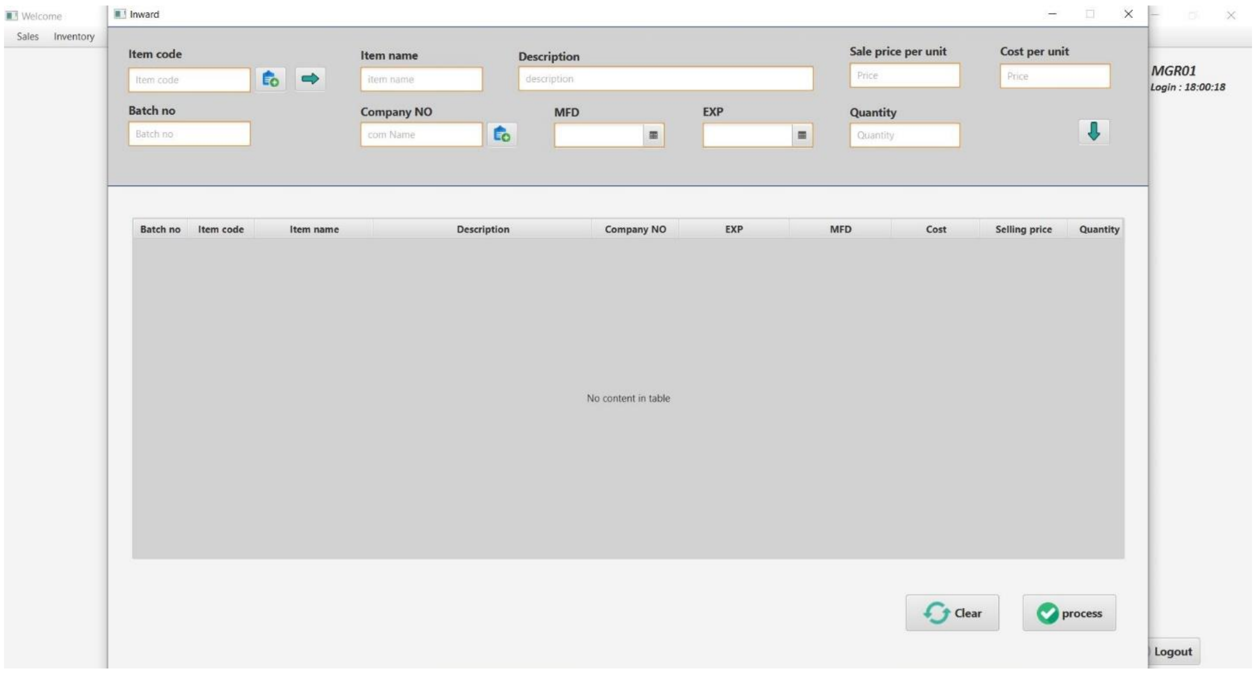


This function gives the user the ability to perform 3 subfunctions.

- Using “Stock Inward” function the user can input stocks and new drugs to the system.
- Using “Stock Outward” function the user could remove stocks of already existing drugs to the system
- Using “Tracker” function the user can track the existing drugs and their stocks in the system

If the user choose the Stock inward function;

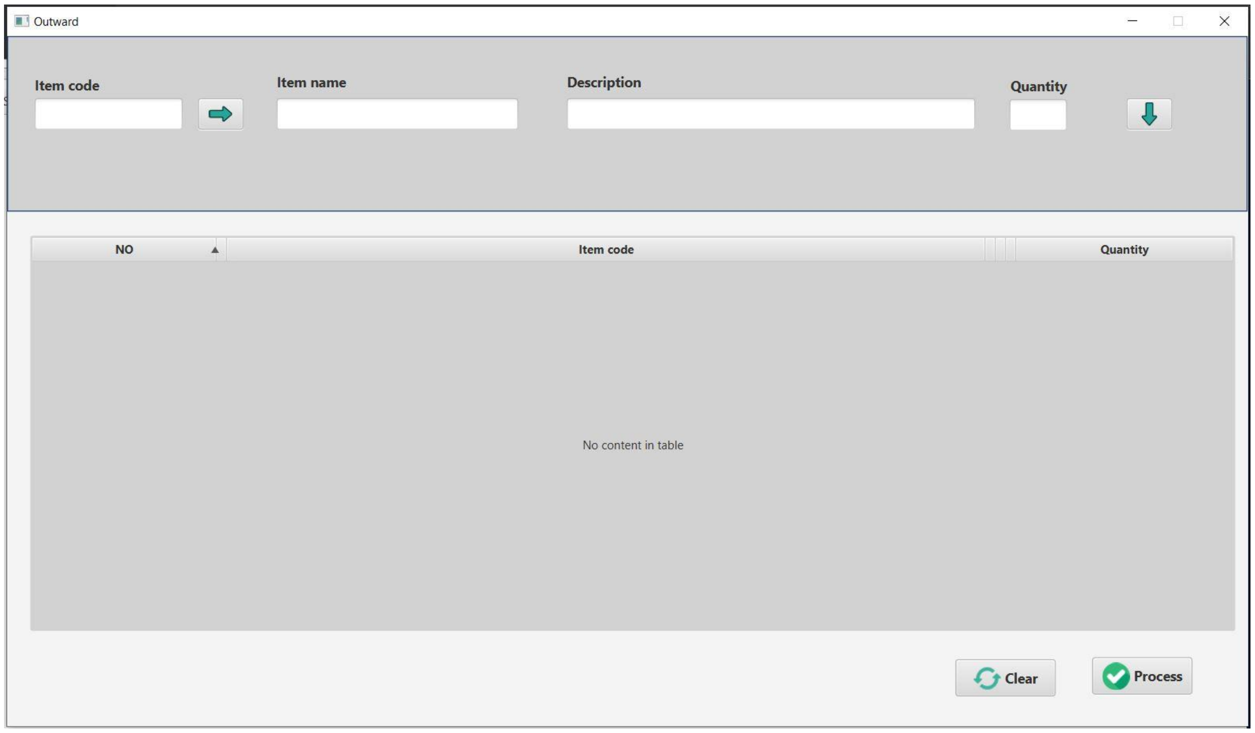
He/she will be directed to following the FXML interface:



Then the user can type the Item Code and press the green arrow if the drug already exists(registered) in the database the Item name and Description will be filled. If these fields are not automatically filled that means the drug is not in the system and a new drug can be added to the system using the add drug button (the button adjacent to the green arrow) and this adds a new entry in to the drug table .

If the user choose the Stock outward function;

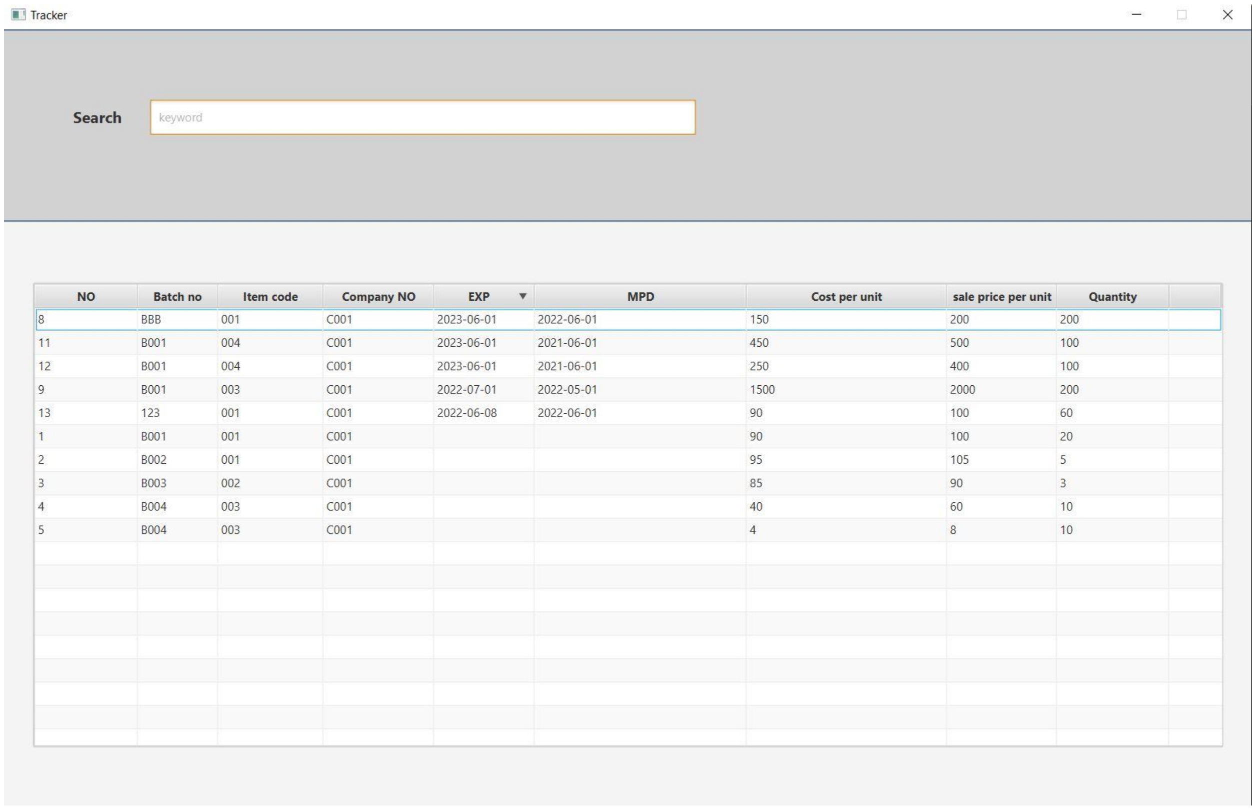
He/she will be directed to following the FXML interface:



In this interface the Item Code can be filled by the user and the Item name, and the Description will be automatically filled. The user can fill the quantity and by pressing the down arrow these entries are added to the following table. This can be finalized by pressing the process key. The quantity in the drug table will be manipulated according to the quantities entered by the user.

If the user choose the Tracker function;

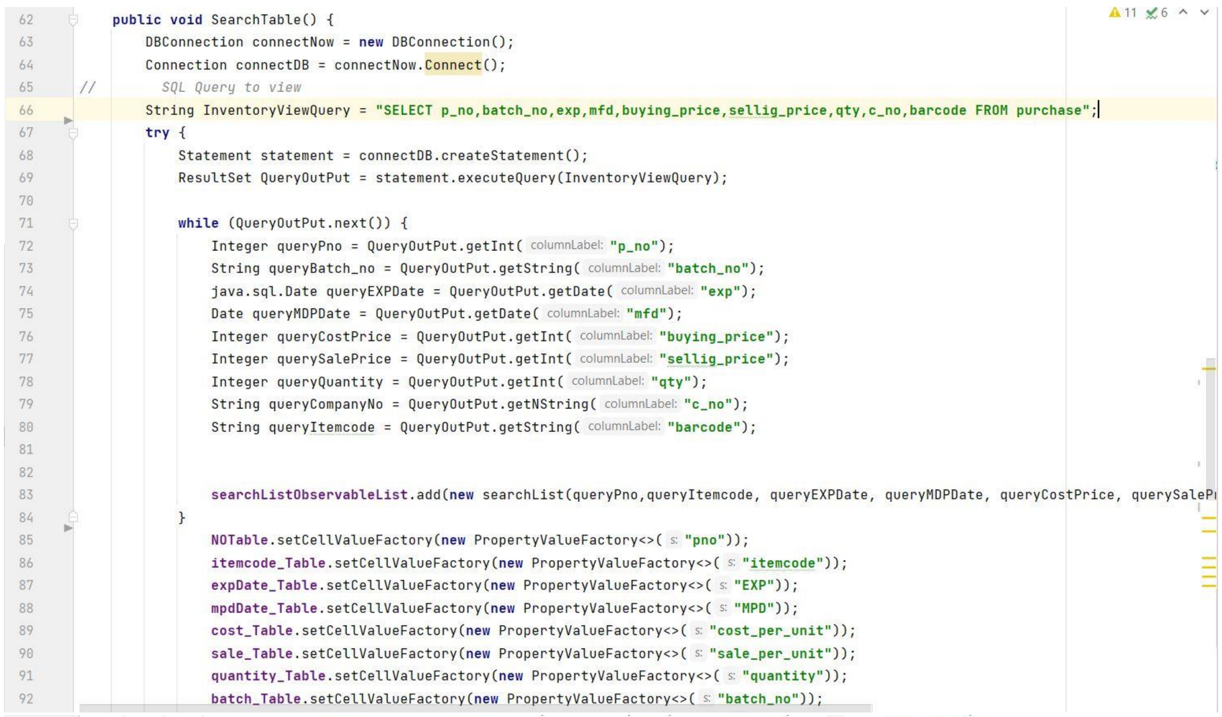
He/she will be directed to following the FXML interface:



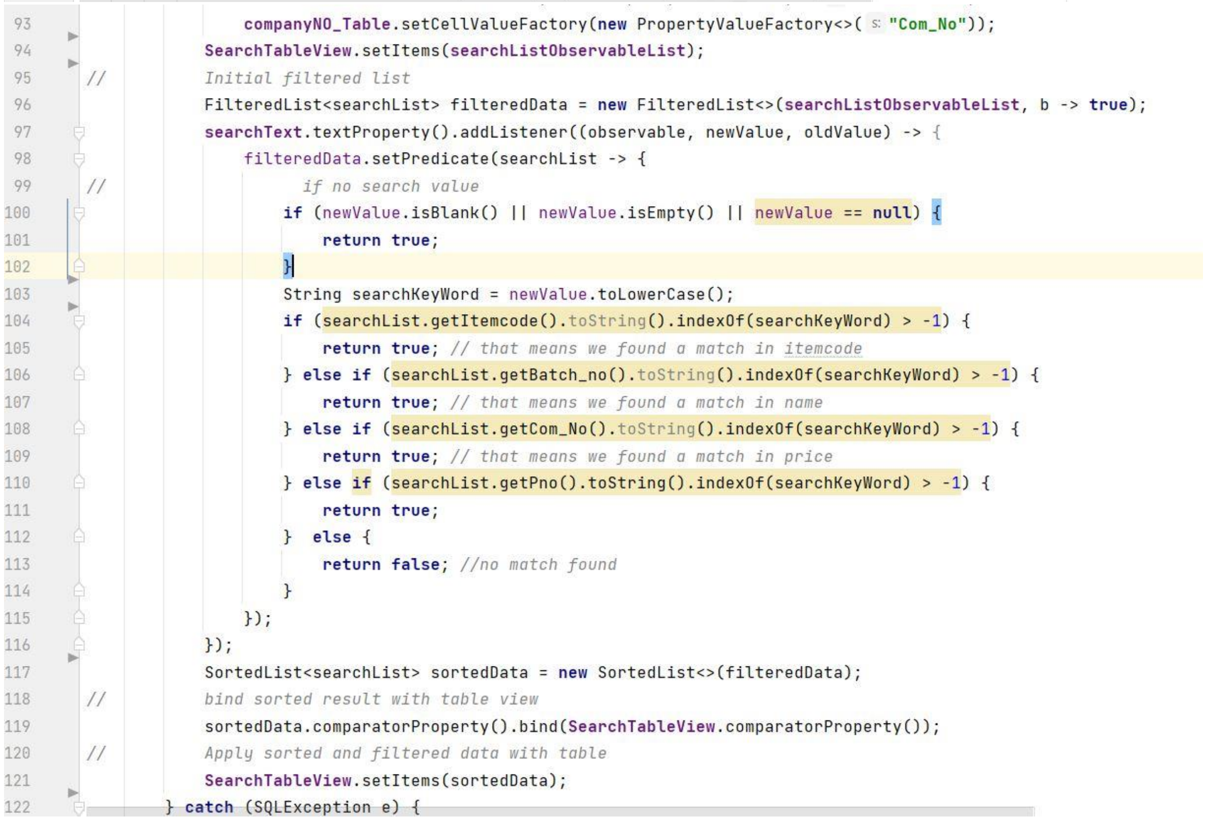
NO	Batch no	Item code	Company NO	EXP ▼	MPD	Cost per unit	sale price per unit	Quantity
8	BBB	001	C001	2023-06-01	2022-06-01	150	200	200
11	B001	004	C001	2023-06-01	2021-06-01	450	500	100
12	B001	004	C001	2023-06-01	2021-06-01	250	400	100
9	B001	003	C001	2022-07-01	2022-05-01	1500	2000	200
13	123	001	C001	2022-06-08	2022-06-01	90	100	60
1	B001	001	C001			90	100	20
2	B002	001	C001			95	105	5
3	B003	002	C001			85	90	3
4	B004	003	C001			40	60	10
5	B004	003	C001			4	8	10

This interface retrieves data from drug and purchase tables and display the quantities present in the stock at the given time.

The search field can be used to sort the items from this list. This functionality of the system is carried out by the following code lines:



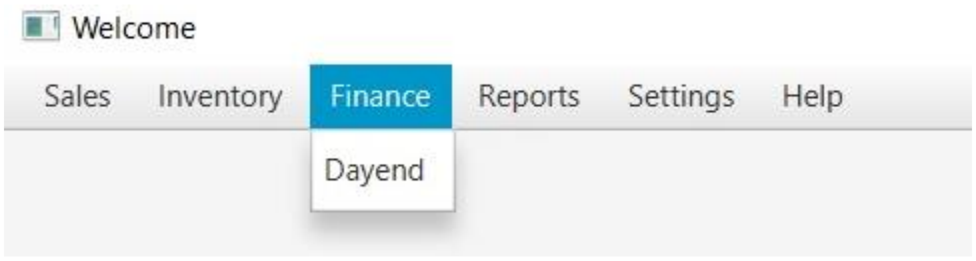
```java
    public void SearchTable() {
        DBConnection connectNow = new DBConnection();
        Connection connectDB = connectNow.Connect();
//        SQL Query to view
        String InventoryViewQuery = "SELECT p_no,batch_no,exp,mfd,buying_price,sellig_price,qty,c_no,barcode FROM purchase";
        try {
            Statement statement = connectDB.createStatement();
            ResultSet QueryOutPut = statement.executeQuery(InventoryViewQuery);

            while (QueryOutPut.next()) {
                Integer queryPno = QueryOutPut.getInt( columnLabel: "p_no");
                String queryBatch_no = QueryOutPut.getString( columnLabel: "batch_no");
                java.sql.Date queryEXPDate = QueryOutPut.getDate( columnLabel: "exp");
                Date queryMDPDate = QueryOutPut.getDate( columnLabel: "mfd");
                Integer queryCostPrice = QueryOutPut.getInt( columnLabel: "buying_price");
                Integer querySalePrice = QueryOutPut.getInt( columnLabel: "sellig_price");
                Integer queryQuantity = QueryOutPut.getInt( columnLabel: "qty");
                String queryCompanyNo = QueryOutPut.getNString( columnLabel: "c_no");
                String queryItemcode = QueryOutPut.getString( columnLabel: "barcode");


                searchListObservableList.add(new searchList(queryPno,queryItemcode, queryEXPDate, queryMDPDate, queryCostPrice, querySalePr
            }
                NOTable.setCellValueFactory(new PropertyValueFactory<>( s: "pno"));
                itemcode_Table.setCellValueFactory(new PropertyValueFactory<>( s: "itemcode"));
                expDate_Table.setCellValueFactory(new PropertyValueFactory<>( s: "EXP"));
                mpdDate_Table.setCellValueFactory(new PropertyValueFactory<>( s: "MPD"));
                cost_Table.setCellValueFactory(new PropertyValueFactory<>( s: "cost_per_unit"));
                sale_Table.setCellValueFactory(new PropertyValueFactory<>( s: "sale_per_unit"));
                quantity_Table.setCellValueFactory(new PropertyValueFactory<>( s: "quantity"));
                batch_Table.setCellValueFactory(new PropertyValueFactory<>( s: "batch_no"));
```



```java
                companyNO_Table.setCellValueFactory(new PropertyValueFactory<>( s: "Com_No"));
            SearchTableView.setItems(searchListObservableList);
//            Initial filtered list
            FilteredList<searchList> filteredData = new FilteredList<>(searchListObservableList, b -> true);
            searchText.textProperty().addListener((observable, newValue, oldValue) -> {
                filteredData.setPredicate(searchList -> {
//                    if no search value
                    if (newValue.isBlank() || newValue.isEmpty() || newValue == null) {
                        return true;
                    }
                    String searchKeyWord = newValue.toLowerCase();
                    if (searchList.getItemcode().toString().indexOf(searchKeyWord) > -1) {
                        return true; // that means we found a match in itemcode
                    } else if (searchList.getBatch_no().toString().indexOf(searchKeyWord) > -1) {
                        return true; // that means we found a match in name
                    } else if (searchList.getCom_No().toString().indexOf(searchKeyWord) > -1) {
                        return true; // that means we found a match in price
                    } else if (searchList.getPno().toString().indexOf(searchKeyWord) > -1) {
                        return true;
                    }  else {
                        return false; //no match found
                    }
                });
            });
            SortedList<searchList> sortedData = new SortedList<>(filteredData);
//            bind sorted result with table view
            sortedData.comparatorProperty().bind(SearchTableView.comparatorProperty());
//            Apply sorted and filtered data with table
            SearchTableView.setItems(sortedData);
        } catch (SQLException e) {
```

The search field can be filled by the user and all the entries in the drug and purchase tables are retrieved to an observable list that “keyword” is compared with that list. And matching entries will be sorted and displayed in the tracker interface this is done by the above code.

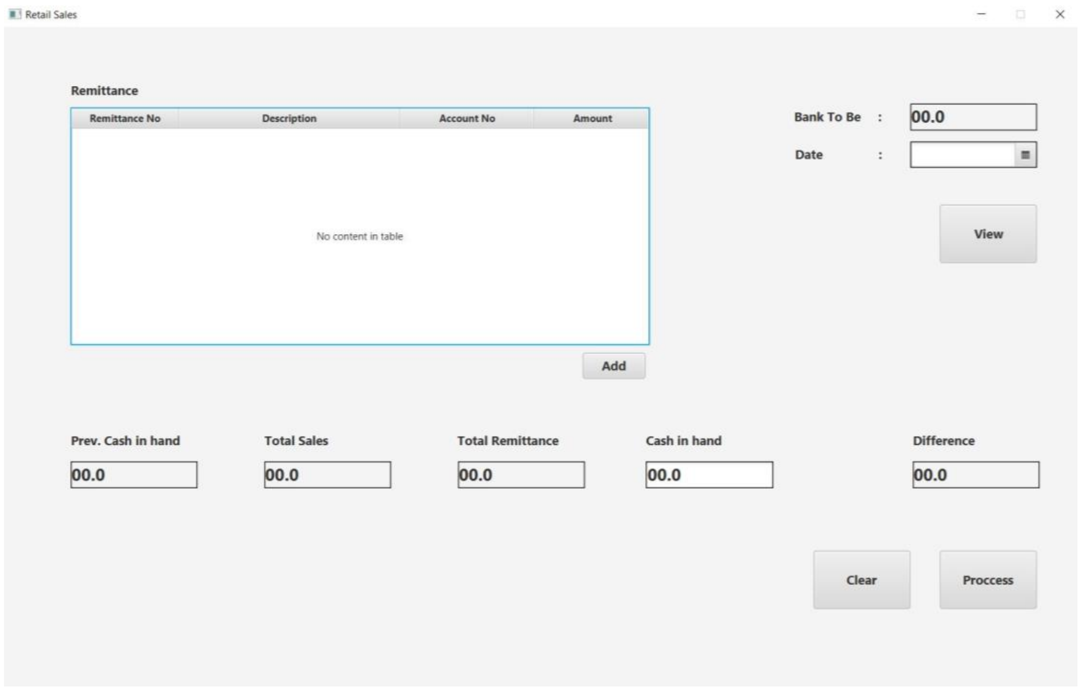
If the user chooses “Finance” function, he/she will be directed to the following menu bar ;



This function gives the user the ability to perform a single sub function;

- Using “Day end” function the user can finalize the days cash from.

When the user choose the Day end function he/she will be directed to the following FXML interface.



Bank To Be :	00.0
Date :	

Prev. Cash in hand	Total Sales	Total Remittance	Cash in hand	Difference
00.0	00.0	00.0	00.0	00.0

The user should select a date and press the view button. Then the previous cash in hand field and the total sales fields will be automatically filled. The difference field will be automatically filled as per the following equation,

$$Difference = (Previous\ day\ cash\ in\ hand + Total\ sales) - (Net\ remitance + Cash\ in\ hand)$$

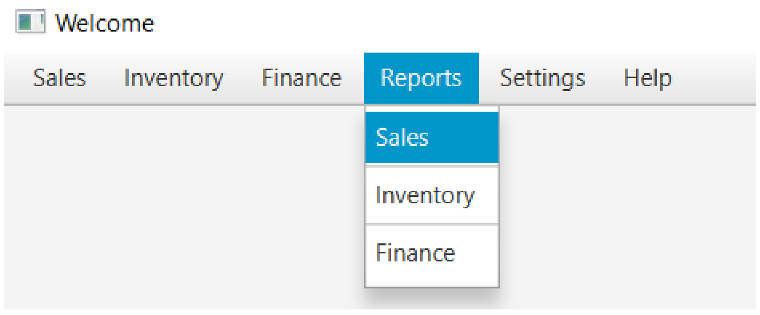
The user can add cash outflows from the pharmacy to the remittance table and then he/she can add the cash amount that is in hand at the end of the day. After these inputs the difference amount will be automatically calculated, and the user can complete the day-end process by checking the cash in hand with the system.

The code structure behind the above functionality is as follows:
The controller class of the above FXML mainly tracks the events of view button and process button.

Once the event of the view button is tracked, the data from the dayend table (dayend table is updated after every single cash sale) and remittance table (remittance table is updated by this interface itself) Then the above-mentioned process is carried out.

Once the event of the process button is tracked, the calculations are conducted again and the final values will be sent to the dayend table. So that it can be used to calculate the cash in hand for the next day.

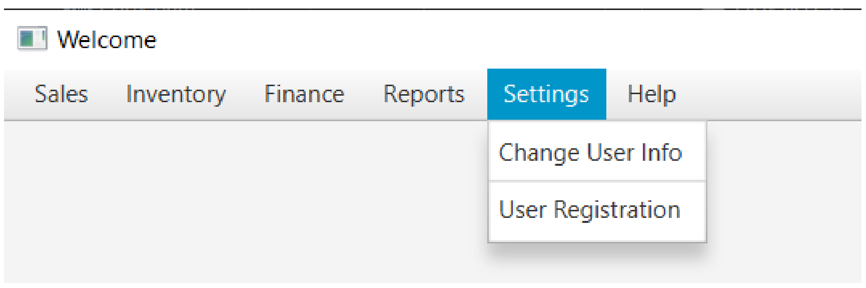
If the user chooses the "Reports" function, he/she will be directed to the following menu bar ;



This function gives the user the ability to perform a several sub functions;

- Even though there are three functionalities in this group. These will be developed in the next stage of the system. (Providing spread sheets and PDF documents will be taken into consideration in the next stage)

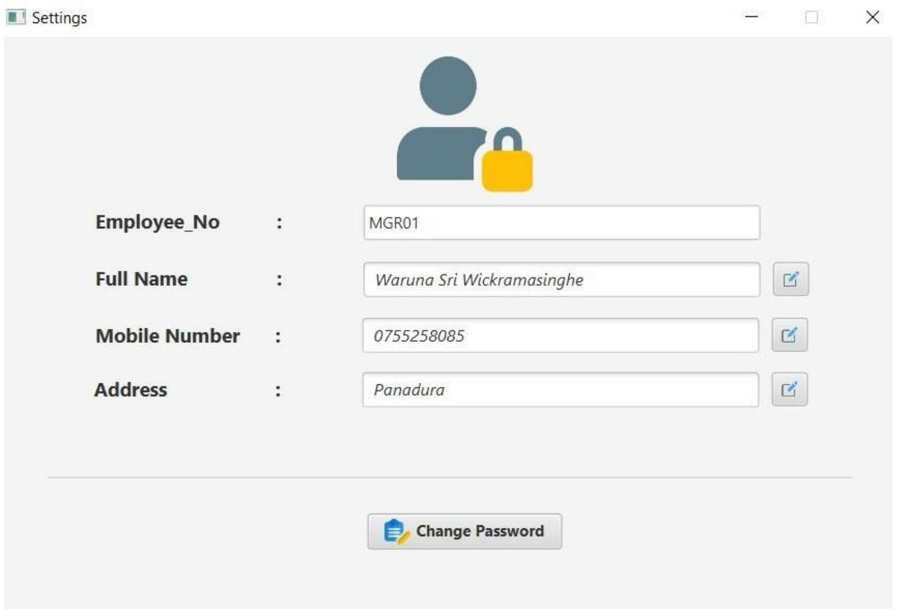
If the user chooses "Settings" function, he/she will be directed to the following menu bar;



This function gives the user the ability to perform 2 sub functions;

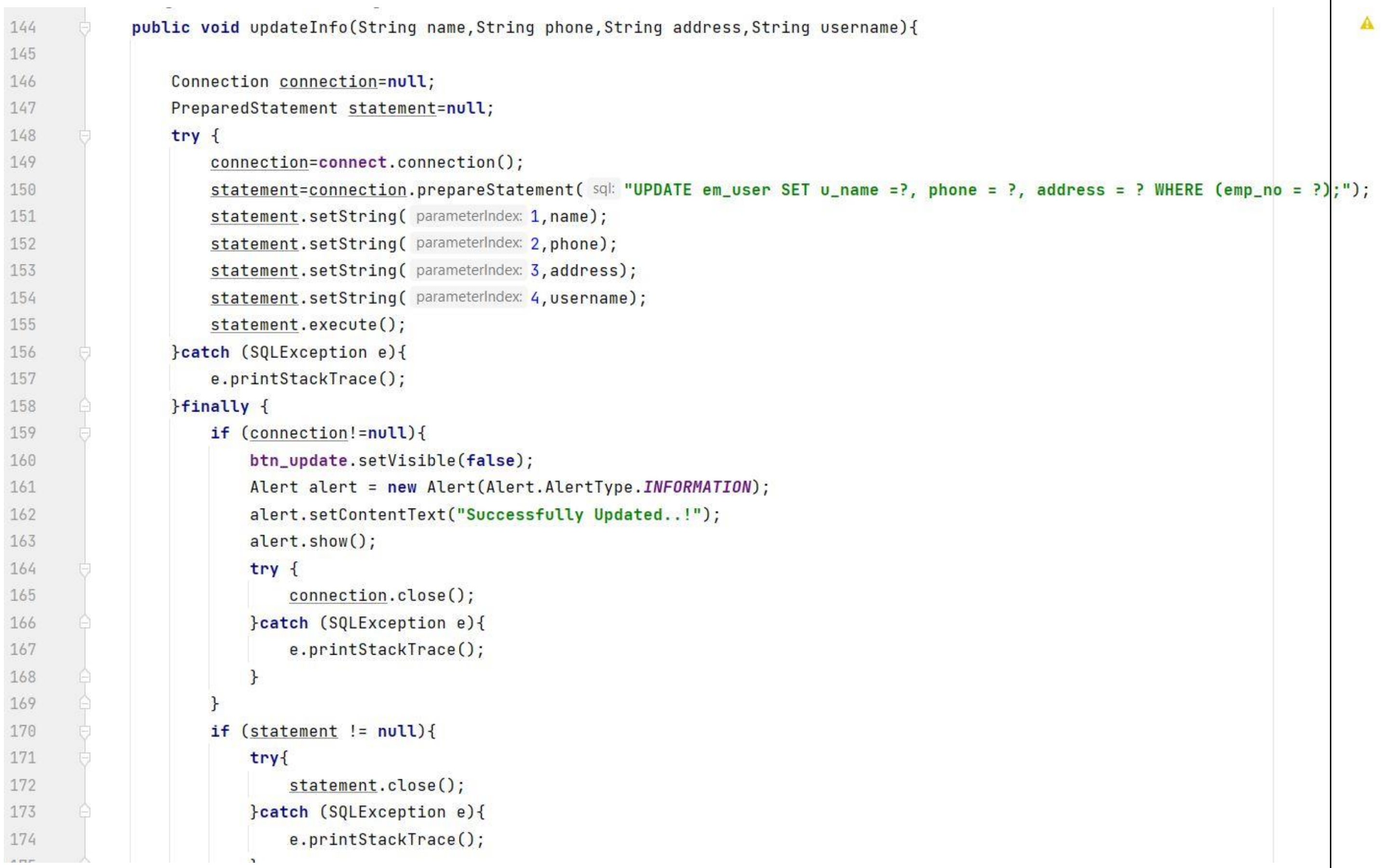
- Using "Change User info" function the user can change his personal information and the password.
- Using the "User Registration" function the user can add new employees into the system.(As a future development access to this functionality will be limited to specific individuals)

When the user choose the "Change User Info" function he/she will be directed in to the following FXML screen;



Employee_No	:	MGR01
Full Name	:	Waruna Sri Wickramasinghe
Mobile Number	:	0755258085
Address	:	Panadura

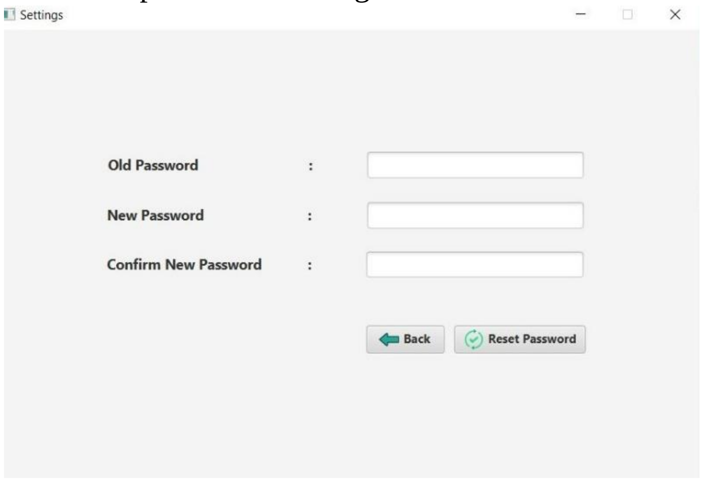
These fields are non editable yet by selecting the edit icon in the right corner of each field the user can edit his/her own details. Then an update button will be visible at the bottom of the interface and the user can save his/her changes by pressing the update button. The code lines associated with this functionality of the system are shown below.



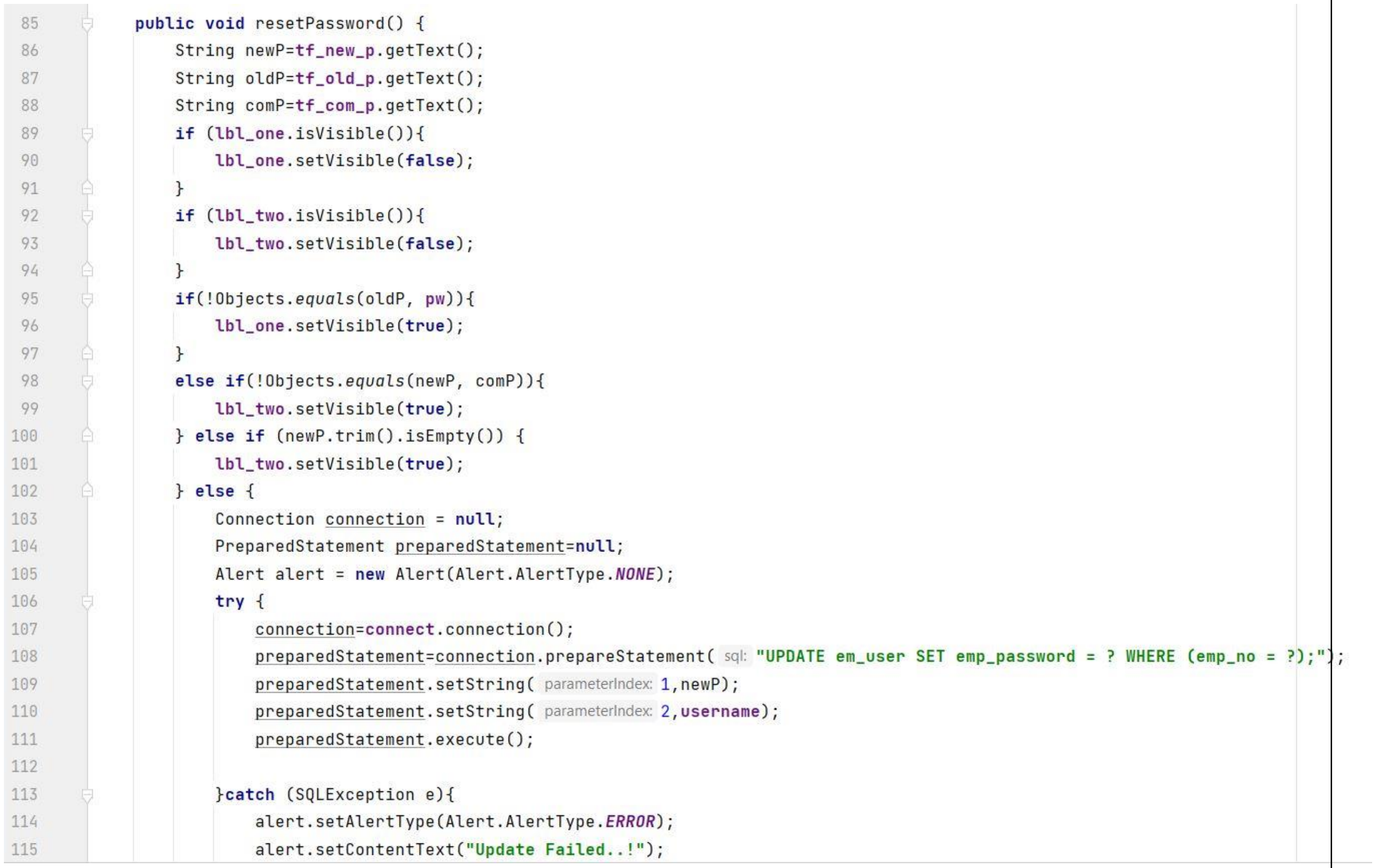
```
public void updateInfo(String name,String phone,String address,String username){

    Connection connection=null;
    PreparedStatement statement=null;
    try {
        connection=connect.connection();
        statement=connection.prepareStatement( sql: "UPDATE em_user SET u_name =?, phone = ?, address = ? WHERE (emp_no = ?);");
        statement.setString( parameterIndex: 1,name);
        statement.setString( parameterIndex: 2,phone);
        statement.setString( parameterIndex: 3,address);
        statement.setString( parameterIndex: 4,username);
        statement.execute();
    }catch (SQLException e){
        e.printStackTrace();
    }finally {
        if (connection!=null){
            btn_update.setVisible(false);
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setContentText("Successfully Updated..!");
            alert.show();
            try {
                connection.close();
            }catch (SQLException e){
                e.printStackTrace();
            }
        }
        if (statement != null){
            try{
                statement.close();
            }catch (SQLException e){
                e.printStackTrace();
```

Once the controller class detect a click on edit button, that will make the fields editable. Once the user finalizes the data will be edited in the em_user table via the “updateInfo” method.

If the user presses the Change Password button he will be directed to the following FXML screen;

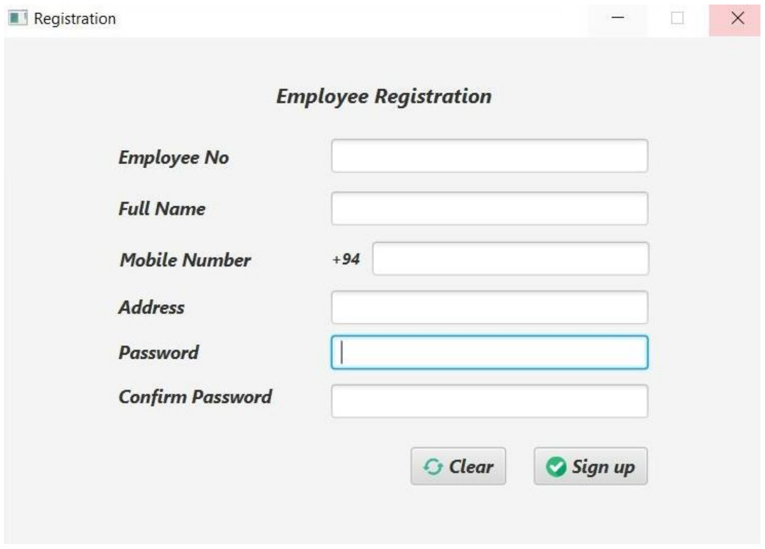


The old password that the user enter will be compared with the already existing password in the em_user table. And if that validation is true user could change the password. After that the new password and the confirmed password will be compared and then comparison the data under the respective user in the em_user table will be modified. This entry is setup where null values will not be accepted to the new password. This mechanism in the system is computed by the following code lines.

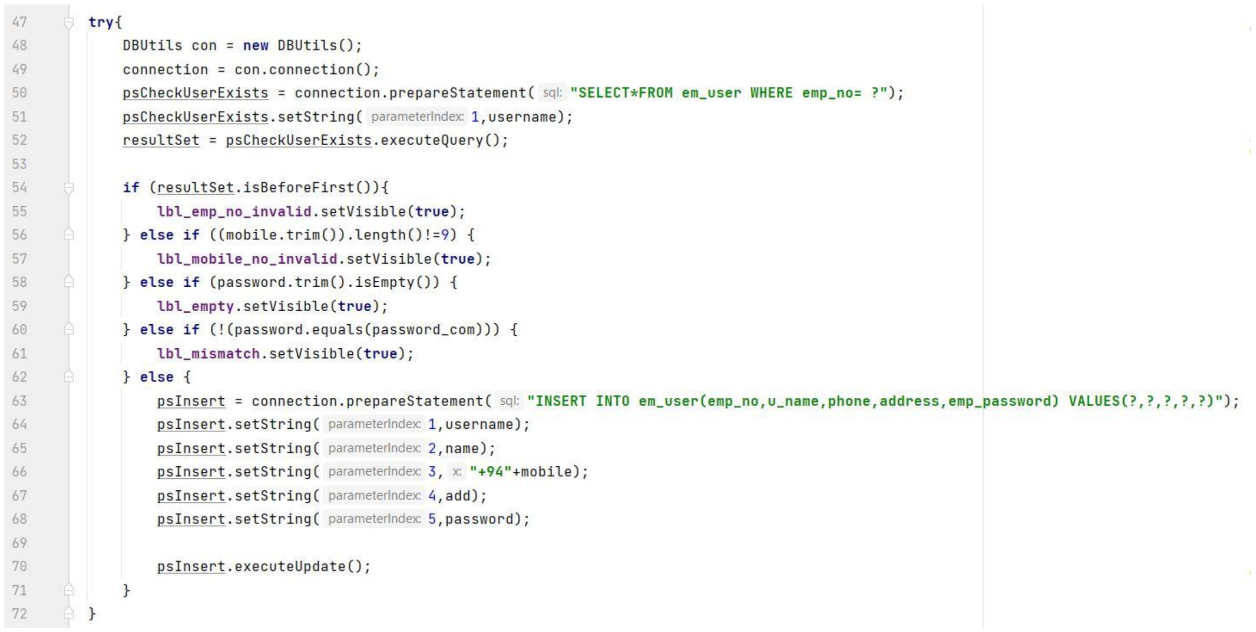


```
public void resetPassword() {
    String newP=tf_new_p.getText();
    String oldP=tf_old_p.getText();
    String comP=tf_com_p.getText();
    if (lbl_one.isVisible()){
        lbl_one.setVisible(false);
    }
    if (lbl_two.isVisible()){
        lbl_two.setVisible(false);
    }
    if(!Objects.equals(oldP, pw)){
        lbl_one.setVisible(true);
    }
    else if(!Objects.equals(newP, comP)){
        lbl_two.setVisible(true);
    } else if (newP.trim().isEmpty()) {
        lbl_two.setVisible(true);
    } else {
        Connection connection = null;
        PreparedStatement preparedStatement=null;
        Alert alert = new Alert(Alert.AlertType.NONE);
        try {
            connection=connect.connection();
            preparedStatement=connection.prepareStatement( sql: "UPDATE em_user SET emp_password = ? WHERE (emp_no = ?);");
            preparedStatement.setString( parameterIndex: 1,newP);
            preparedStatement.setString( parameterIndex: 2,username);
            preparedStatement.execute();

        }catch (SQLException e){
            alert.setAlertType(Alert.AlertType.ERROR);
            alert.setContentText("Update Failed..!");
```

When the user choose the “User Registration” function he/she will be directed in to the following FXML screen;

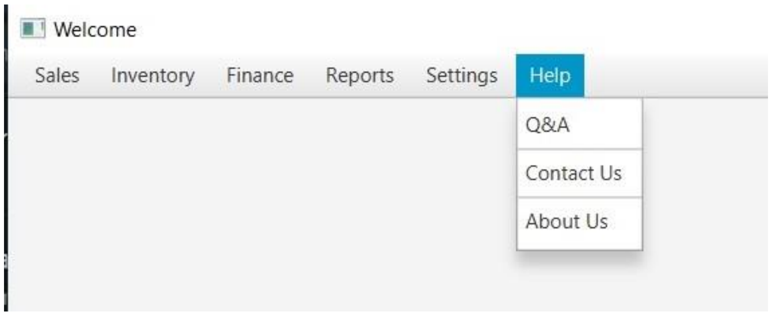


The users can insert new entries in to the em_user table using this interface. After a user inputs a new entry(details of a new employee) to the system the system will validate the new employee number with the existing entries in the emp_no column thus it is used as a primary key. The same process is conducted while setting up a password for the new employee apart from the validation of the existing password. The following code lines are the structure behind the above process.



```
try{
    DBUtils con = new DBUtils();
    connection = con.connection();
    psCheckUserExists = connection.prepareStatement( sql: "SELECT*FROM em_user WHERE emp_no= ?");
    psCheckUserExists.setString( parameterIndex: 1,username);
    resultSet = psCheckUserExists.executeQuery();

    if (resultSet.isBeforeFirst()){
        lbl_emp_no_invalid.setVisible(true);
    } else if ((mobile.trim()).length()!=9) {
        lbl_mobile_no_invalid.setVisible(true);
    } else if (password.trim().isEmpty()) {
        lbl_empty.setVisible(true);
    } else if (!(password.equals(password_com))) {
        lbl_mismatch.setVisible(true);
    } else {
        psInsert = connection.prepareStatement( sql: "INSERT INTO em_user(emp_no,u_name,phone,address,emp_password) VALUES(?,?,?,?,?)");
        psInsert.setString( parameterIndex: 1,username);
        psInsert.setString( parameterIndex: 2,name);
        psInsert.setString( parameterIndex: 3, x: "+94"+mobile);
        psInsert.setString( parameterIndex: 4,add);
        psInsert.setString( parameterIndex: 5,password);

        psInsert.executeUpdate();
    }
}
```

If the user chooses “Help” function, he/she will be directed to the following menu bar ;



This function gives the user the ability to perform 3 sub functions;

- By pressing “Q & A” tab the user can get answers to the questions that are faced by the users when handling the system daily.
- By pressing “Contact Us” tab the user can get connected to the developing team.
- By pressing the “About Us” tab the user can get details about the developing team.

Future Developments

- Generating of reports.(By enabling spreadsheets and PDFs)
- Improve the access control.
- Cancellation of sales and purchases.
- Enhance the validation process
    - When inputting user details(Access will be controlled to managers)
    - When inputting drug details(Access will be controlled to pharmacists)
    - When inputting user details(Access will be controlled to financial people in the pharmacy(cashiers))
- Connecting the system into a centralized server and providing multiple user access.
- Track the user interaction with the system using a log.
- Include the barcode scanning function to the system.

Team Members

- Nihim Abhayarathne - IM/2018/099
- Waruna Sri Wickramasinghe - IM/2019/028
- Mathurshi Vijayarajendran - IM/2019/037
- Naveen Jayathilaka - IM/2019/044
- Vishaka Bandara - IM/2019/046
- Tharindu Adhikari - IM/2019/052
- Tharindu Herath - IM/2019/055
- Udara Suranimala - IM/2019/080
- Nipuni De Silva - IM/2019/093
- Lashadya Dasanyake - IM/2019/103
- Sandushi Weraduwa - IM/2019/112