**common**

package common;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import model.Cashier;

import model.Sale;

public class Common {

public static Cashier cashierrec = new Cashier();

public static double totalAmount ;

public static double change;

public static double payamount;

public static int purchaseid;

public static ObservableList<String> buygetdata = FXCollections.observableArrayList();

public static ObservableList<Sale> saleitemsdatafromsaletable = FXCollections.observableArrayList();

public static String productids;

public static String productqtys;

public static String slipno;

public static String paidtype = "";

public static String buygetpromo = "";

public static String buygetitem = "";

public static String cardinfo ="";

public static boolean isIntegratedDevice = false;

}

**controller**

**AdminCardController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ResourceBundle;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import model.CardUser;

public class AdminCardController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private TableView<CardUser> tb\_card;

@FXML

private JFXTextField tf\_card\_no;

@FXML

private JFXTextField tf\_pin;

@FXML

private JFXButton bt\_new;

@FXML

private JFXButton bt\_add;

private TableColumn<CardUser, String> col\_item\_card\_no;

private TableColumn<CardUser, String> col\_item\_amount;

private TableColumn<CardUser, String> col\_item\_pin;

private TableColumn<CardUser, String> col\_item\_customer;

private TableColumn<CardUser, String> col\_item\_lastused\_date;

private TableColumn<CardUser, String> col\_item\_registerdate;

private TableColumn<CardUser, String> col\_item\_expiredate;

private ObservableList<CardUser> cardData = FXCollections.observableArrayList();

@FXML

void initialize() throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

assert tb\_card != null : "fx:id=\"tb\_card\" was not injected: check your FXML file 'Admin\_card.fxml'.";

assert tf\_card\_no != null : "fx:id=\"tf\_card\_no\" was not injected: check your FXML file 'Admin\_card.fxml'.";

assert tf\_pin != null : "fx:id=\"tf\_pin\" was not injected: check your FXML file 'Admin\_card.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Admin\_card.fxml'.";

assert bt\_add != null : "fx:id=\"bt\_add\" was not injected: check your FXML file 'Admin\_card.fxml'.";

tf\_card\_no.setEditable(false);

tf\_pin.setEditable(false);

col\_item\_card\_no = new TableColumn<CardUser, String>("Card Number");

col\_item\_amount = new TableColumn<CardUser, String>("Balance Amount");

col\_item\_pin = new TableColumn<CardUser, String>("PIN");

col\_item\_customer = new TableColumn<CardUser, String>("Customer ID");

col\_item\_lastused\_date = new TableColumn<CardUser, String>("Last Used Date");

col\_item\_registerdate = new TableColumn<CardUser, String>("Register Date");

col\_item\_expiredate = new TableColumn<CardUser, String>("Expired Date");

col\_item\_card\_no.setMinWidth(200.0);

col\_item\_amount.setMinWidth(200.0);

col\_item\_pin.setMinWidth(200.0);

col\_item\_customer.setMinWidth(200.0);

col\_item\_lastused\_date.setMinWidth(200.0);

col\_item\_registerdate.setMinWidth(200.0);

col\_item\_expiredate.setMinWidth(200.0);

col\_item\_card\_no.setStyle("-fx-font-size: 18");

col\_item\_amount.setStyle("-fx-font-size: 18");

col\_item\_pin.setStyle("-fx-font-size: 18");

col\_item\_customer.setStyle("-fx-font-size: 18");

col\_item\_lastused\_date.setStyle("-fx-font-size: 18");

col\_item\_registerdate.setStyle("-fx-font-size: 18");

col\_item\_expiredate.setStyle("-fx-font-size: 18");

col\_item\_card\_no.setCellValueFactory(new PropertyValueFactory<CardUser, String>("cardno"));

col\_item\_amount.setCellValueFactory(new PropertyValueFactory<CardUser, String>("amount"));

col\_item\_pin.setCellValueFactory(new PropertyValueFactory<CardUser, String>("pin"));

col\_item\_customer.setCellValueFactory(new PropertyValueFactory<CardUser, String>("customrid"));

col\_item\_lastused\_date.setCellValueFactory(new PropertyValueFactory<CardUser, String>("lastdateused"));

col\_item\_registerdate.setCellValueFactory(new PropertyValueFactory<CardUser, String>("registerdate"));

col\_item\_expiredate.setCellValueFactory(new PropertyValueFactory<CardUser, String>("expireddate"));

tb\_card.getColumns().addAll(col\_item\_card\_no, col\_item\_customer, col\_item\_amount, col\_item\_pin,

col\_item\_lastused\_date, col\_item\_registerdate, col\_item\_expiredate);

// get data from db

String query = "SELECT \* FROM `Card`";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while(rs.next()) {

CardUser c = new CardUser();

c.setCardno(rs.getString(1));

c.setCustomrid(rs.getString(2));

c.setAmount(rs.getString(3));

c.setLastdateused(rs.getString(4));

c.setRegisterdate(rs.getString(5));

c.setExpireddate(rs.getString(6));

c.setPin(""+rs.getInt(7));

cardData.add(c);

}

//set data to table

tb\_card.setItems(cardData);

//row double click action

tb\_card.setRowFactory(t -> {

TableRow<CardUser> row = new TableRow<>();

row.setOnMouseClicked(e -> {

//get data from selected row

if (e.getClickCount() == 2 && (! row.isEmpty()) ) {

CardUser ca = tb\_card.getSelectionModel().getSelectedItem();

System.out.println("Double click is: "+ca.getCardno());

//set data to tf

tf\_card\_no.setText(ca.getCardno());

tf\_pin.setText(ca.getPin());

}

});

return row;

});

}

@FXML

void onAddAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

if(tf\_card\_no.getText().equals("") || tf\_pin.getText().equals("")) {

Alert al = new Alert(AlertType.ERROR, "Invalid Input or Data Missing!");

al.showAndWait();

}

else {

String cardno = tf\_card\_no.getText().toString();

String getpin = tf\_pin.getText().toString();

try {

String queryAdd = "INSERT INTO `Card`(`cardnumber`, `customerid`, `amount`, `lastuseddate`, `registerdate`, `expireddate`, `pin`) "

+ "VALUES ('"+cardno+"','','0','','','',"+getpin+")";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(queryAdd);

//clear data array

cardData.clear();

//get data set set to db

// get data from db

String query = "SELECT \* FROM `Card`";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while(rs.next()) {

CardUser c = new CardUser();

c.setCardno(rs.getString(1));

c.setCustomrid(rs.getString(2));

c.setAmount(rs.getString(3));

c.setLastdateused(rs.getString(4));

c.setRegisterdate(rs.getString(5));

c.setExpireddate(rs.getString(6));

c.setPin(""+rs.getInt(7));

cardData.add(c);

}

//set data to table

tb\_card.refresh();

//clear tf

tf\_card\_no.clear();

tf\_pin.clear();

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item added!");

al.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}//end of else

}

@FXML

void onNewAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

tf\_card\_no.clear();

tf\_pin.clear();

String query = "SELECT Card.cardnumber FROM `Card` ORDER BY Card.cardnumber DESC LIMIT 1";

String oldid = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsold = DBInitialize.statement.executeQuery(query);

while(rsold.next()) {

oldid = ""+rsold.getString(1);

}

//count +1 new Id

String newId = ""+(Long.parseLong(oldid) + 1 );

tf\_card\_no.setText(newId);

//generate a 4 digit integer 1000 <10000

int pin = (int)(Math.random()\*9000)+1000;

tf\_pin.setText(""+pin);

}

}

**AdminCashierController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXRadioButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.MenuItem;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.ToggleGroup;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import model.Cashier;

public class AdminCashierController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private TableView<Cashier> tb\_cashier;

private TableColumn<Cashier, String> col\_item\_id;

private TableColumn<Cashier, String> col\_item\_name;

private TableColumn<Cashier, String> col\_item\_age;

private TableColumn<Cashier, String> col\_item\_gender;

private TableColumn<Cashier, String> col\_item\_addr;

private TableColumn<Cashier, String> col\_item\_phone;

private TableColumn<Cashier, String> col\_item\_email;

private TableColumn<Cashier, String> col\_item\_password;

private TableColumn<Cashier, String> col\_item\_date\_created;

private ObservableList<Cashier> cashierData = FXCollections.observableArrayList();

@FXML

private JFXButton bt\_cashier\_add;

@FXML

private JFXButton bt\_cashier\_update;

@FXML

private JFXTextField tf\_id;

@FXML

private JFXTextField tf\_name;

@FXML

private JFXTextField tf\_age;

@FXML

private JFXRadioButton rdo\_male;

@FXML

private ToggleGroup gender;

@FXML

private JFXRadioButton rdo\_female;

@FXML

private JFXTextField tf\_addr;

@FXML

private JFXTextField tf\_phone;

@FXML

private JFXTextField tf\_email;

@FXML

private JFXTextField tf\_password;

@FXML

private JFXTextField tf\_desc;

@FXML

private JFXTextField tf\_date\_created;

@FXML

private JFXButton bt\_new;

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tb\_cashier != null : "fx:id=\"tb\_cashier\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert bt\_cashier\_add != null : "fx:id=\"bt\_cashier\_add\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert bt\_cashier\_update != null : "fx:id=\"bt\_cashier\_update\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_id != null : "fx:id=\"tf\_id\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_age != null : "fx:id=\"tf\_age\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert rdo\_male != null : "fx:id=\"rdo\_male\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert gender != null : "fx:id=\"gender\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert rdo\_female != null : "fx:id=\"rdo\_female\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_addr != null : "fx:id=\"tf\_addr\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_phone != null : "fx:id=\"tf\_phone\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_email != null : "fx:id=\"tf\_email\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_password != null : "fx:id=\"tf\_password\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert tf\_date\_created != null : "fx:id=\"tf\_date\_created\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Admin\_cashier.fxml'.";

tf\_date\_created.setEditable(true);

//set today date

String pattern = "dd/MM/yyyy";

String datecreated =new SimpleDateFormat(pattern).format(new Date());

System.out.println("last date use is "+datecreated);

tf\_date\_created.setEditable(false);

tf\_date\_created.setText(datecreated);

col\_item\_id = new TableColumn<Cashier, String>("ID");

col\_item\_name = new TableColumn<Cashier, String>("Name");

col\_item\_age = new TableColumn<Cashier, String>("Age");

col\_item\_gender = new TableColumn<Cashier, String>("Gender");

col\_item\_addr = new TableColumn<Cashier, String>("Address");

col\_item\_phone = new TableColumn<Cashier, String>("Phone");

col\_item\_email = new TableColumn<Cashier, String>("Email");

col\_item\_password = new TableColumn<Cashier, String>("Password");

col\_item\_date\_created = new TableColumn<Cashier, String>("Date Created");

col\_item\_id.setMinWidth(90.0);

col\_item\_name.setMinWidth(180.0);

col\_item\_age.setMinWidth(50.0);

col\_item\_gender.setMinWidth(70.0);

col\_item\_addr.setMinWidth(230.0);

col\_item\_phone.setMinWidth(200.0);

col\_item\_email.setMinWidth(280.0);

col\_item\_password.setMinWidth(150.0);

col\_item\_date\_created.setMinWidth(120.0);

col\_item\_id.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_age.setStyle("-fx-font-size: 18");

col\_item\_gender.setStyle("-fx-font-size: 18");

col\_item\_addr.setStyle("-fx-font-size: 18");

col\_item\_phone.setStyle("-fx-font-size: 18");

col\_item\_email.setStyle("-fx-font-size: 18");

col\_item\_password.setStyle("-fx-font-size: 18");

col\_item\_date\_created.setStyle("-fx-font-size: 18");

col\_item\_id.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("id"));

col\_item\_name.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("name"));

col\_item\_age.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("age"));

col\_item\_gender.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("gender"));

col\_item\_addr.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("address"));

col\_item\_phone.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("phone"));

col\_item\_email.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("email"));

col\_item\_password.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("password"));

col\_item\_date\_created.setCellValueFactory(

new PropertyValueFactory<Cashier, String>("dateCreated"));

tb\_cashier.getColumns().addAll(col\_item\_id

,col\_item\_name

,col\_item\_age

,col\_item\_gender

,col\_item\_addr

,col\_item\_phone

,col\_item\_email

,col\_item\_password

,col\_item\_date\_created);

//get data in cashier table from database

String query = "SELECT \* FROM `cashier` ORDER BY cashier.id DESC;";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

Cashier cashier = new Cashier();

cashier.setId(""+rs.getInt(1));

cashier.setName(rs.getString(2));

cashier.setAge(""+rs.getString(3));

cashier.setGender(rs.getString(4));

cashier.setAddress(rs.getString(5));

cashier.setPhone(rs.getString(6));

cashier.setEmail(rs.getString(7));

cashier.setPassword(rs.getString(8));

cashier.setDateCreated(rs.getString(9));

cashierData.add(cashier);

}

//set data to table

tb\_cashier.setItems(cashierData);

tb\_cashier.setRowFactory(t -> {

TableRow<Cashier> row = new TableRow<>();

row.setOnMouseClicked(e -> {

//get data from selected row

if (e.getClickCount() == 2 && (! row.isEmpty()) ) {

Cashier cashier = tb\_cashier.getSelectionModel().getSelectedItem();

System.out.println("Double click is: "+cashier.getName());

//get gender data

String gender = cashier.getGender();

if(gender.equals("male")) {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

}else {

rdo\_female.setSelected(true);

rdo\_male.setSelected(false);

}

//add data to the text field

tf\_id.setText(cashier.getId());

tf\_name.setText(cashier.getName());

tf\_age.setText(cashier.getAge());

tf\_addr.setText(cashier.getAddress());

tf\_phone.setText(cashier.getPhone());

tf\_email.setText(cashier.getEmail());

tf\_password.setText(cashier.getPassword());

tf\_date\_created.setText(cashier.getDateCreated());

}

});

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Cashier ca = tb\_cashier.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION, "Are U Sure To Delete " + ca.getName() + " ?", ButtonType.YES, ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

//do stuff

String removequery = "DELETE FROM `cashier` WHERE cashier.id = '"+ca.getId()+"';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(removequery);

//update table

//update table data

new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT \* FROM `cashier` ORDER BY cashier.id DESC;";

new DBInitialize();

ResultSet rsss = DBInitialize.statement.executeQuery(queryupdatetable);

cashierData.clear();

while(rsss.next()) {

Cashier cashier = new Cashier();

cashier.setId(""+rsss.getInt(1));

cashier.setName(rsss.getString(2));

cashier.setAge(""+rsss.getString(3));

cashier.setGender(rsss.getString(4));

cashier.setAddress(rsss.getString(5));

cashier.setPhone(rsss.getString(6));

cashier.setEmail(rsss.getString(7));

cashier.setPassword(rsss.getString(8));

cashier.setDateCreated(rsss.getString(9));

cashierData.add(cashier);

}

//tb\_product\_item.getItems().clear();

tb\_cashier.refresh();

/\*//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item removed!");

al.showAndWait();\*/

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item deleted!");

al.showAndWait();

} catch (ClassNotFoundException | SQLException | InstantiationException | IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

});

rowMenu.getItems().addAll( removeItem);

// only display context menu for non-null items:

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty()))

.then(rowMenu)

.otherwise((ContextMenu)null));

return row;

});

}

@FXML

void onbtNewAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

//clear

tf\_id.clear();

tf\_name.clear();

tf\_age.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_email.clear();

tf\_password.clear();

tf\_date\_created.clear();

String getoldidquery = "SELECT cashier.id FROM cashier ORDER BY cashier.id DESC LIMIT 1";

String olCadid = "";

System.out.println("old id is "+olCadid);

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsgetid = DBInitialize.statement.executeQuery(getoldidquery);

while(rsgetid.next()) {

olCadid = ""+rsgetid.getInt(1);

}

System.out.println("old id is "+olCadid);

//count +1 new Id

String newCaId = ""+(Integer.parseInt(olCadid) + 1 );

tf\_id.setText(newCaId);

System.out.println("new id is "+newCaId);

//create today date

String pattern = "dd/MM/yyyy";

String todaydate =new SimpleDateFormat(pattern).format(new Date());

System.out.println("today is "+todaydate);

tf\_date\_created.setText(todaydate);

}

@FXML

void onBtAddAction(ActionEvent event) throws SQLException {

if(tf\_name.getText().equals("") || tf\_id.getText().equals("") || tf\_age.getText().equals("") || tf\_addr.getText().equals("") || tf\_phone.getText().equals("") || tf\_email.getText().equals("") || tf\_password.getText().equals("")

|| tf\_date\_created.getText().equals("") || tf\_age.getText().matches(".\*[a-zA-Z]+.\*") || Integer.parseInt(tf\_age.getText()) < 18 || tf\_phone.getText().matches(".\*[a-zA-Z]+.\*") || tf\_phone.getText().length() < 9 || !tf\_email.getText().contains("mail.com") || !tf\_email.getText().contains("@")) {

Alert al = new Alert(AlertType.ERROR, "Invalid Input or Data Missing!");

al.showAndWait();

}

else {

int id = Integer.parseInt(tf\_id.getText().toString());

String name = tf\_name.getText().toString();

String age = tf\_age.getText().toString();

String addr = tf\_addr.getText().toString();

String ph = tf\_phone.getText().toString();

String mail = tf\_email.getText().toString();

String pw = tf\_password.getText().toString();

String datecreated = tf\_date\_created.getText().toString();

String gender = "";

if(rdo\_male.isSelected()) {

gender = "male";

}else {

gender = "female";

}

try {

String addQuery = "INSERT INTO `cashier` (`id`, `name`, `age`, `gender`, `address`, `phone`, `email`, `password`, `date created`) VALUES ( "+id+", '"+name+"', "+age+", '"+gender+"','"+addr+"','"+ph+"','"+mail+"','"+pw+"','"+datecreated+"')";

new DBInitialize();

DBInitialize.statement.executeUpdate(addQuery);

//clear

tf\_id.clear();

tf\_name.clear();

tf\_age.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_email.clear();

tf\_password.clear();

tf\_date\_created.clear();

//get tabe data

String getTableDataQuery = "SELECT \* FROM `cashier` ORDER BY cashier.id DESC;";

cashierData.clear();//clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

Cashier ca = new Cashier();

ca.setId(getrs.getString(1));

ca.setName(getrs.getString(2));

ca.setAge(getrs.getString(3));

ca.setGender(getrs.getString(4));

ca.setAddress(getrs.getString(5));

ca.setPhone(getrs.getString(6));

ca.setEmail(getrs.getString(7));

ca.setPassword(getrs.getString(8));

ca.setDateCreated(getrs.getString(9));

cashierData.add(ca);

}

//set to table

tb\_cashier.refresh();

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item added!");

al.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}//end of else

}

@FXML

void onbtUpdateAction(ActionEvent event) throws SQLException {

if(tf\_name.getText().equals("") || tf\_id.getText().equals("") || tf\_age.getText().equals("") || tf\_addr.getText().equals("") || tf\_phone.getText().equals("") || tf\_email.getText().equals("") || tf\_password.getText().equals("")

|| tf\_date\_created.getText().equals("") || tf\_age.getText().matches(".\*[a-zA-Z]+.\*") || Integer.parseInt(tf\_age.getText()) < 18 || tf\_phone.getText().matches(".\*[a-zA-Z]+.\*") || tf\_phone.getText().length() < 9 || !tf\_email.getText().contains("mail.com") || !tf\_email.getText().contains("@")) {

Alert al = new Alert(AlertType.ERROR, "Invalid Input or Data Missing!");

al.showAndWait();

}

else {

int id = Integer.parseInt(tf\_id.getText().toString());

String name = tf\_name.getText().toString();

String age = tf\_age.getText().toString();

String addr = tf\_addr.getText().toString();

String ph = tf\_phone.getText().toString();

String mail = tf\_email.getText().toString();

String pw = tf\_password.getText().toString();

String datecreated = tf\_date\_created.getText().toString();

String gender = "";

if(rdo\_male.isSelected()) {

gender = "male";

}else {

gender = "female";

}

String updateQuery = "UPDATE `cashier` SET `name`='"+name+"',`age`='"+age+"',`gender`='"+gender+"',"

+ "`address`='"+addr+"',`phone`='"+ph+"',`email`='"+mail+"',`password`='"+pw+"',`date created`='"+datecreated+"' WHERE `id`= "+id+";";

new DBInitialize();

DBInitialize.statement.executeUpdate(updateQuery);

//clear

tf\_id.clear();

tf\_name.clear();

tf\_age.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_email.clear();

tf\_password.clear();

tf\_date\_created.clear();

//get tabe data

String getTableDataQuery = "SELECT \* FROM `cashier` ORDER BY cashier.id DESC;";

cashierData.clear();//clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

Cashier ca = new Cashier();

ca.setId(getrs.getString(1));

ca.setName(getrs.getString(2));

ca.setAge(getrs.getString(3));

ca.setGender(getrs.getString(4));

ca.setAddress(getrs.getString(5));

ca.setPhone(getrs.getString(6));

ca.setEmail(getrs.getString(7));

ca.setPassword(getrs.getString(8));

ca.setDateCreated(getrs.getString(9));

cashierData.add(ca);

}

//set to table

tb\_cashier.refresh();

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item updated!");

al.showAndWait();

}//end of if

}

}

**AdminCategoryController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.MenuItem;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import model.Category;

public class AdminCategoryController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private TableView<Category> tb\_category;

private TableColumn<Category, String> col\_item\_id;

private TableColumn<Category, String> col\_item\_name;

private TableColumn<Category, String> col\_item\_date\_created;

private ObservableList<Category> categoryData = FXCollections.observableArrayList();

@FXML

private JFXTextField tf\_id;

@FXML

private JFXTextField tf\_name;

@FXML

private JFXTextField tf\_date\_created;

@FXML

private JFXButton bt\_new;

@FXML

private JFXButton bt\_add;

@FXML

private JFXButton bt\_update;

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tb\_category != null : "fx:id=\"tb\_category\" was not injected: check your FXML file 'Admin\_category.fxml'.";

assert tf\_id != null : "fx:id=\"tf\_id\" was not injected: check your FXML file 'Admin\_category.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'Admin\_category.fxml'.";

assert tf\_date\_created != null : "fx:id=\"tf\_date\_created\" was not injected: check your FXML file 'Admin\_category.fxml'.";

assert bt\_add != null : "fx:id=\"bt\_add\" was not injected: check your FXML file 'Admin\_category.fxml'.";

assert bt\_update != null : "fx:id=\"bt\_update\" was not injected: check your FXML file 'Admin\_category.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Admin\_category.fxml'.";

tf\_date\_created.setEditable(true);

//current date

String pattern = "dd/MM/yyyy";

String datecreated =new SimpleDateFormat(pattern).format(new Date());

System.out.println("last date use is "+datecreated);

tf\_date\_created.setText(datecreated);

col\_item\_id = new TableColumn<Category, String>("ID");

col\_item\_name = new TableColumn<Category, String>("Name");

col\_item\_date\_created = new TableColumn<Category, String>("Date Created");

col\_item\_id.setMinWidth(150.0);

col\_item\_name.setMinWidth(280.0);

col\_item\_date\_created.setMinWidth(200.0);

col\_item\_id.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_date\_created.setStyle("-fx-font-size: 18");

col\_item\_id.setCellValueFactory(

new PropertyValueFactory<Category, String>("id"));

col\_item\_name.setCellValueFactory(

new PropertyValueFactory<Category, String>("name"));

col\_item\_date\_created.setCellValueFactory(

new PropertyValueFactory<Category, String>("dateCreated"));

tb\_category.getColumns().addAll(col\_item\_id

,col\_item\_name

,col\_item\_date\_created

);

//setup

tf\_id.clear();

tf\_name.clear();

tf\_date\_created.clear();

String querygetId = "SELECT `id` FROM `productcategory` ORDER BY productcategory.id DESC LIMIT 1";

String oldid = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rss = DBInitialize.statement.executeQuery(querygetId);

while(rss.next()) {

oldid = ""+rss.getInt(1);

}

//count +1 new Id

String newId = ""+(Integer.parseInt(oldid) + 1 );

tf\_id.setText(newId);

//create today date

String pattern1 = "dd/MM/yyyy";

String todaydate =new SimpleDateFormat(pattern1).format(new Date());

System.out.println("today is "+todaydate);

tf\_date\_created.setText(todaydate);

//---------------------------------------------------------------------

//get data from db and add to category model

String query= "SELECT \* FROM `productcategory`";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

Category ca = new Category();

ca.setId(""+rs.getInt(1));

ca.setName(rs.getString(2));

ca.setDateCreated(rs.getString(3));

categoryData.add(ca);

}

//add data to table

tb\_category.setItems(categoryData);

//row double click action

tb\_category.setRowFactory(t -> {

TableRow<Category> row = new TableRow<>();

row.setOnMouseClicked(e -> {

//get data from selected row

if (e.getClickCount() == 2 && (! row.isEmpty()) ) {

Category category = tb\_category.getSelectionModel().getSelectedItem();

System.out.println("Double click is: "+category.getName());

//set data to tf

tf\_id.setText(category.getId());

tf\_name.setText(category.getName());

tf\_date\_created.setText(category.getDateCreated());

}

});

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Category ca = tb\_category.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION, "Are U Sure To Delete " + ca.getName() + " ?", ButtonType.YES, ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

//check if this current category is used in product items

//get count

int cateCount = 0;

String getCateUsedCount = "SELECT COUNT(\*) FROM productitems, productcategory WHERE productcategory.id = productitems.categoryid AND productcategory.id = '"+ca.getId()+"'";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsCateCount = DBInitialize.statement.executeQuery(getCateUsedCount);

while(rsCateCount.next()) {

cateCount = rsCateCount.getInt(1);

}

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException

| SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

if(cateCount > 0 ) {

//show can't delete Alert

Alert aal = new Alert(AlertType.ERROR, "Cannot Delete! This Category has been used in product items. You can delete the product items that link with this category and try again. Thanks!");

aal.showAndWait();

}else {

//do stuff

String removequery = "DELETE FROM `productcategory` WHERE productcategory.id = '"+ca.getId()+"';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(removequery);

//update table

//update table data

new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT \* FROM `productcategory`;";

new DBInitialize();

ResultSet rsss = DBInitialize.statement.executeQuery(queryupdatetable);

categoryData.clear();

while(rsss.next()) {

Category caa = new Category();

caa.setId(""+rsss.getInt(1));

caa.setName(rsss.getString(2));

caa.setDateCreated(rsss.getString(3));

categoryData.add(caa);

}

//tb\_product\_item.getItems().clear();

tb\_category.refresh();

/\*//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item removed!");

al.showAndWait();\*/

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item deleted!");

al.showAndWait();

} catch (ClassNotFoundException | SQLException | InstantiationException | IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

});

rowMenu.getItems().addAll( removeItem);

// only display context menu for non-null items:

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty()))

.then(rowMenu)

.otherwise((ContextMenu)null));

return row;

});

}

@FXML

void onbtnewAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

tf\_id.clear();

tf\_name.clear();

tf\_date\_created.clear();

String query = "SELECT `id` FROM `productcategory` ORDER BY productcategory.id DESC LIMIT 1";

String oldid = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while(rs.next()) {

oldid = ""+rs.getInt(1);

}

//count +1 new Id

String newId = ""+(Integer.parseInt(oldid) + 1 );

tf\_id.setText(newId);

//create today date

String pattern = "dd/MM/yyyy";

String todaydate =new SimpleDateFormat(pattern).format(new Date());

System.out.println("today is "+todaydate);

tf\_date\_created.setText(todaydate);

}

@FXML

void onAddAction(ActionEvent event) throws SQLException {

if(tf\_id.getText().equals("") || tf\_name.getText().equals("") || tf\_date\_created.getText().equals("") ) {

Alert al = new Alert(AlertType.ERROR, "Invaild Input or Data Missing!");

al.showAndWait();

}

else {

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String datecreated = tf\_date\_created.getText().toString();

try {

String addQuery = "INSERT INTO `productcategory`(`id`, `name`, `datecreated`) VALUES ("+Integer.parseInt(id)+", '"+name+"', '"+datecreated+"')";

new DBInitialize();

DBInitialize.statement.executeUpdate(addQuery);

tf\_id.clear();

tf\_name.clear();

tf\_date\_created.clear();

try {

//get tabe data

String getTableDataQuery = "SELECT \* FROM `productcategory`;";

categoryData.clear();//clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

Category ca = new Category();

ca.setId(""+getrs.getInt(1));

ca.setName(getrs.getString(2));

ca.setDateCreated(getrs.getString(3));

categoryData.add(ca);

}

//set to table

tb\_category.refresh();

//tb\_category.setItems(categoryData);

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item added!");

al.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}//end of else of if

}

@FXML

void onUpdateAction(ActionEvent event) throws SQLException {

if(tf\_id.getText().equals("") || tf\_name.getText().equals("") || tf\_date\_created.getText().equals("") ) {

Alert al = new Alert(AlertType.ERROR, "Invaild Input or Data Missing!");

al.showAndWait();

}

else {

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String datecreated = tf\_date\_created.getText().toString();

String updateQuery = "UPDATE `productcategory` SET `name`='"+name+"',`datecreated`='"+datecreated+"' WHERE `id`="+id+";";

new DBInitialize();

DBInitialize.statement.executeUpdate(updateQuery);

tf\_id.clear();

tf\_name.clear();

tf\_date\_created.clear();

//get tabe data

String getTableDataQuery = "SELECT \* FROM `productcategory`;";

categoryData.clear();//clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

Category ca = new Category();

ca.setId(""+getrs.getInt(1));

ca.setName(getrs.getString(2));

ca.setDateCreated(getrs.getString(3));

categoryData.add(ca);

}

//set to table

tb\_category.refresh();

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item updated!");

al.showAndWait();

}

}//end of first if

}

**AdminCustomerController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXRadioButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.time.format.DateTimeFormatter;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.MenuItem;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.ToggleGroup;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import model.CardUser;

public class AdminCustomerController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private TableView<CardUser> tb\_customer;

@FXML

private JFXTextField tf\_id;

@FXML

private JFXTextField tf\_name;

@FXML

private JFXTextField tf\_addr;

@FXML

private JFXTextField tf\_phone;

@FXML

private JFXTextField tf\_mail;

@FXML

private JFXTextField tf\_card\_no;

@FXML

private JFXTextField tf\_amount;

@FXML

private JFXTextField tf\_last\_date\_used;

@FXML

private JFXTextField tf\_date\_created;

@FXML

private JFXTextField tf\_pin;

@FXML

private JFXTextField tf\_age;

@FXML

private JFXButton bt\_search;

@FXML

private JFXRadioButton rdo\_male;

@FXML

private ToggleGroup gender;

@FXML

private JFXRadioButton rdo\_female;

@FXML

private JFXButton bt\_new;

@FXML

private JFXTextField tf\_expired\_date;

@FXML

private JFXButton bt\_add;

@FXML

private JFXButton bt\_update;

private TableColumn<CardUser, String> col\_item\_id;

private TableColumn<CardUser, String> col\_item\_name;

private TableColumn<CardUser, String> col\_item\_age;

private TableColumn<CardUser, String> col\_item\_gender;

private TableColumn<CardUser, String> col\_item\_addr;

private TableColumn<CardUser, String> col\_item\_phone;

private TableColumn<CardUser, String> col\_item\_email;

private TableColumn<CardUser, String> col\_item\_card\_no;

private TableColumn<CardUser, String> col\_item\_amount;

private TableColumn<CardUser, String> col\_item\_last\_date\_used;

private TableColumn<CardUser, String> col\_item\_expired\_date;

private TableColumn<CardUser, String> col\_item\_date\_created;

private TableColumn<CardUser, String> col\_item\_pin;

private ObservableList<CardUser> customerData = FXCollections.observableArrayList();

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tb\_customer != null : "fx:id=\"tb\_customer\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_id != null : "fx:id=\"tf\_id\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_age != null : "fx:id=\"tf\_age\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_addr != null : "fx:id=\"tf\_addr\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_phone != null : "fx:id=\"tf\_phone\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_mail != null : "fx:id=\"tf\_mail\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_card\_no != null : "fx:id=\"tf\_card\_no\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_amount != null : "fx:id=\"tf\_amount\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_last\_date\_used != null : "fx:id=\"tf\_last\_date\_used\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_pin != null : "fx:id=\"tf\_pin\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_date\_created != null : "fx:id=\"tf\_date\_created\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert rdo\_male != null : "fx:id=\"rdo\_male\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert gender != null : "fx:id=\"gender\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert rdo\_female != null : "fx:id=\"rdo\_female\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert bt\_add != null : "fx:id=\"bt\_add\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert bt\_update != null : "fx:id=\"bt\_update\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert bt\_search != null : "fx:id=\"bt\_search\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

assert tf\_expired\_date != null : "fx:id=\"tf\_expired\_date\" was not injected: check your FXML file 'Admin\_customer.fxml'.";

tf\_pin.setEditable(false);

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

System.out.println("last date use is " + todaydate);

tf\_date\_created.setText(todaydate);

// count expired date to 3 years

String[] todaydateAry = todaydate.split("/");

String day = todaydateAry[0];

String month = todaydateAry[1];

String year = todaydateAry[2];

year = "" + (Integer.parseInt(year) + 3);

tf\_expired\_date.setText(day+"/"+month+"/" + year);

col\_item\_id = new TableColumn<CardUser, String>("ID");

col\_item\_name = new TableColumn<CardUser, String>("Name");

col\_item\_age = new TableColumn<CardUser, String>("Age");

col\_item\_gender = new TableColumn<CardUser, String>("Gender");

col\_item\_addr = new TableColumn<CardUser, String>("Address");

col\_item\_phone = new TableColumn<CardUser, String>("Phone");

col\_item\_email = new TableColumn<CardUser, String>("Email");

col\_item\_card\_no = new TableColumn<CardUser, String>("Card Number");

col\_item\_amount = new TableColumn<CardUser, String>("Amount");

col\_item\_last\_date\_used = new TableColumn<CardUser, String>("Last Date Used");

col\_item\_expired\_date = new TableColumn<CardUser, String>("Expired Date");

col\_item\_date\_created = new TableColumn<CardUser, String>("Date Created");

col\_item\_pin = new TableColumn<CardUser, String>("PIN");

col\_item\_id.setMinWidth(90.0);

col\_item\_name.setMinWidth(200.0);

col\_item\_age.setMinWidth(75.0);

col\_item\_gender.setMinWidth(90.0);

col\_item\_addr.setMinWidth(130.0);

col\_item\_phone.setMinWidth(150.0);

col\_item\_email.setMinWidth(200.0);

col\_item\_card\_no.setMinWidth(200.0);

col\_item\_amount.setMinWidth(110.0);

col\_item\_last\_date\_used.setMinWidth(110.0);

col\_item\_expired\_date.setMinWidth(110.0);

col\_item\_date\_created.setMinWidth(110.0);

col\_item\_pin.setMinWidth(90.0);

col\_item\_id.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_age.setStyle("-fx-font-size: 18");

col\_item\_gender.setStyle("-fx-font-size: 18");

col\_item\_addr.setStyle("-fx-font-size: 18");

col\_item\_phone.setStyle("-fx-font-size: 18");

col\_item\_email.setStyle("-fx-font-size: 18");

col\_item\_card\_no.setStyle("-fx-font-size: 18");

col\_item\_amount.setStyle("-fx-font-size: 18");

col\_item\_last\_date\_used.setStyle("-fx-font-size: 18");

col\_item\_expired\_date.setStyle("-fx-font-size: 18");

col\_item\_date\_created.setStyle("-fx-font-size: 18");

col\_item\_pin.setStyle("-fx-font-size: 18");

col\_item\_id.setCellValueFactory(new PropertyValueFactory<CardUser, String>("customrid"));

col\_item\_name.setCellValueFactory(new PropertyValueFactory<CardUser, String>("name"));

col\_item\_age.setCellValueFactory(new PropertyValueFactory<CardUser, String>("age"));

col\_item\_gender.setCellValueFactory(new PropertyValueFactory<CardUser, String>("gender"));

col\_item\_addr.setCellValueFactory(new PropertyValueFactory<CardUser, String>("address"));

col\_item\_phone.setCellValueFactory(new PropertyValueFactory<CardUser, String>("phone"));

col\_item\_email.setCellValueFactory(new PropertyValueFactory<CardUser, String>("email"));

col\_item\_card\_no.setCellValueFactory(new PropertyValueFactory<CardUser, String>("cardno"));

col\_item\_amount.setCellValueFactory(new PropertyValueFactory<CardUser, String>("amount"));

col\_item\_last\_date\_used.setCellValueFactory(new PropertyValueFactory<CardUser, String>("lastdateused"));

col\_item\_expired\_date.setCellValueFactory(new PropertyValueFactory<CardUser, String>("expireddate"));

col\_item\_date\_created.setCellValueFactory(new PropertyValueFactory<CardUser, String>("registerdate"));

col\_item\_pin.setCellValueFactory(new PropertyValueFactory<CardUser, String>("pin"));

tb\_customer.getColumns().addAll(col\_item\_id, col\_item\_name, col\_item\_age, col\_item\_gender, col\_item\_addr,

col\_item\_phone, col\_item\_email, col\_item\_card\_no, col\_item\_amount, col\_item\_last\_date\_used,

col\_item\_date\_created, col\_item\_expired\_date, col\_item\_pin);

// get data from db and add into array

String query = "SELECT Customer.id, Customer.name, Customer.age, Customer.gender, Customer.address, Customer.phone, Customer.email, Card.cardnumber, Card.amount, Card.lastuseddate, Card.registerdate, Card.expireddate, Card.pin FROM Customer, Card WHERE Customer.id = Card.customerid ORDER BY Customer.id DESC;";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

CardUser cu = new CardUser();

cu.setCustomrid(rs.getString(1));

cu.setName(rs.getString(2));

cu.setAge(rs.getString(3));

cu.setGender(rs.getString(4));

cu.setAddress(rs.getString(5));

cu.setPhone(rs.getString(6));

cu.setEmail(rs.getString(7));

cu.setCardno(rs.getString(8));

cu.setAmount(rs.getString(9));

cu.setLastdateused(rs.getString(10));

cu.setRegisterdate(rs.getString(11));

cu.setExpireddate(rs.getString(12));

cu.setPin(rs.getString(13));

customerData.add(cu);

}

// set data to table

tb\_customer.setItems(customerData);

// row action

tb\_customer.setRowFactory(t -> {

TableRow<CardUser> row = new TableRow<>();

row.setOnMouseClicked(e -> {

// get data from selected row

if (e.getClickCount() == 2 && (!row.isEmpty())) {

CardUser cu = tb\_customer.getSelectionModel().getSelectedItem();

System.out.println("Double click is: " + cu.getName());

// set data to tf

tf\_id.setText(cu.getCustomrid());

tf\_name.setText(cu.getName());

tf\_age.setText(cu.getAge());

tf\_addr.setText(cu.getAddress());

tf\_phone.setText(cu.getPhone());

tf\_mail.setText(cu.getEmail());

tf\_card\_no.setText(cu.getCardno());

tf\_amount.setText(cu.getAmount());

tf\_last\_date\_used.setText(cu.getLastdateused());

tf\_expired\_date.setText(cu.getExpireddate());

tf\_pin.setText(cu.getPin());

tf\_date\_created.setText(cu.getRegisterdate());

// rdo

String gender = cu.getGender();

if (gender.equals("male")) {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

} else {

rdo\_male.setSelected(false);

rdo\_female.setSelected(true);

}

}

});

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

CardUser ca = tb\_customer.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION,

"Are U Sure You Want To Delete " + ca.getName() + " from database including card ?", ButtonType.YES,

ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

// do stuff

String removeCustomerquery = "DELETE FROM `Customer` WHERE Customer.id = '" + ca.getCustomrid()

+ "';";

String removeCardquery = "DELETE FROM `Card` WHERE Card.cardnumber = '" + ca.getCardno() + "';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(removeCustomerquery);

DBInitialize.statement.executeUpdate(removeCardquery);

// update table

// get tabe data

String getTableDataQuery = "SELECT Customer.id, Customer.name, Customer.age, Customer.gender, Customer.address, Customer.phone, Customer.email, Card.cardnumber, Card.amount, Card.lastuseddate, Card.registerdate, Card.expireddate, Card.pin FROM Customer, Card WHERE Customer.id = Card.customerid ORDER BY Customer.id DESC;";

customerData.clear();// clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

CardUser cu = new CardUser();

cu.setCustomrid(getrs.getString(1));

cu.setName(getrs.getString(2));

cu.setAge(getrs.getString(3));

cu.setGender(getrs.getString(4));

cu.setAddress(getrs.getString(5));

cu.setPhone(getrs.getString(6));

cu.setEmail(getrs.getString(7));

cu.setCardno(getrs.getString(8));

cu.setAmount(getrs.getString(9));

cu.setLastdateused(getrs.getString(10));

cu.setRegisterdate(getrs.getString(11));

cu.setExpireddate(getrs.getString(12));

cu.setPin("" + getrs.getInt(13));

customerData.add(cu);

}

// set to table

tb\_customer.refresh();

/\*

\* //show alert Alert al = new Alert(AlertType.INFORMATION, "Item removed!");

\* al.showAndWait();

\*/

// show alert

Alert al = new Alert(AlertType.INFORMATION, "Item deleted!");

al.showAndWait();

} catch (ClassNotFoundException | SQLException | InstantiationException

| IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

});

rowMenu.getItems().addAll(removeItem);

// only display context menu for non-null items:

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty())).then(rowMenu).otherwise((ContextMenu) null));

return row;

});

}

// search card no info

@FXML

void onBtSearchAction(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

String cardno = tf\_card\_no.getText().toString();

// search card by card no and add info to the tf

String searchCardQuery = "SELECT \* FROM `Card` WHERE Card.cardnumber = '" + cardno + "';";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rssearch = DBInitialize.statement.executeQuery(searchCardQuery);

String datecreated = "";

String expireddate = "";

String lastDateUsed = "";

String customerid = "";

if (rssearch.next()) {

expireddate = rssearch.getString("expireddate");

tf\_amount.setText(rssearch.getString("amount"));

lastDateUsed = rssearch.getString("lastuseddate");

datecreated = rssearch.getString("registerdate");

tf\_pin.setText("" + rssearch.getInt("pin"));

customerid = rssearch.getString("customerid");

} else {

// show alert

Alert al = new Alert(AlertType.ERROR, " Invalid Card!");

al.showAndWait();

}

System.out.println("expiredate" + expireddate);

/\*

\* if(datecreated.equals("") || expireddate.equals("") ||

\* lastDateUsed.equals("") ) {

\*

\* String tdy =new SimpleDateFormat(pattern).format(new Date());

\* System.out.println("today tdy is "+tdy);

\*

\*

\* tf\_date\_created.setText(tdy); tf\_expired\_date.setValue(LocalDate.parse(tdy,

\* formatter)); tf\_last\_date\_used.setText(tdy);

\*

\* }else { //tf\_date\_created.setText(datecreated);

\* tf\_expired\_date.setValue(LocalDate.parse(expireddate, formatter));

\* //tf\_last\_date\_used.setText(lastDateUsed);

\*

\* }

\*/

// create today date

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

System.out.println("today is " + todaydate);

// count expired date to 3 years

String[] todaydateAry = todaydate.split("/");

String day = todaydateAry[0];

String month = todaydateAry[1];

String year = todaydateAry[2];

year = "" + (Integer.parseInt(year) + 3);

if (customerid.equals("")) {

// tf\_id.clear();

tf\_name.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_mail.clear();

tf\_age.clear();

tf\_date\_created.setText(todaydate);

// tf\_expired\_date.setValue(LocalDate.parse(todaydate, formatter));

tf\_expired\_date.setText(day+"/"+month+"/" + year);

tf\_last\_date\_used.setText(todaydate);

} else {

String name = "";

String age = "";

String gender = "";

String addr = "";

String ph = "";

String mail = "";

String searchCustomerQuery = "SELECT `name`, `age`, `gender`, `address`, `phone`, `email` FROM `Customer` WHERE Customer.id = '"

+ customerid + "';";

new DBInitialize();

ResultSet cusrs = DBInitialize.statement.executeQuery(searchCustomerQuery);

while (cusrs.next()) {

name = cusrs.getString(1);

age = cusrs.getString(2);

gender = cusrs.getString(3);

addr = cusrs.getString(4);

ph = cusrs.getString(5);

mail = cusrs.getString(6);

}

if (gender.equals("male")) {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

} else {

rdo\_male.setSelected(false);

rdo\_female.setSelected(true);

}

tf\_id.setText(customerid);

tf\_name.setText(name);

tf\_age.setText(age);

tf\_addr.setText(addr);

tf\_phone.setText(ph);

tf\_mail.setText(mail);

// DateTimeFormatter format = DateTimeFormatter.ofPattern("dd/MM/yyyy");

System.out.println("exipredddddddateeee: " + expireddate);

tf\_date\_created.setText(datecreated);

tf\_expired\_date.getText().toString();

tf\_expired\_date.clear();

tf\_expired\_date.setText(expireddate);

tf\_last\_date\_used.setText(lastDateUsed);

}

}

@FXML

void onbtNewAction(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// set data to tf

tf\_id.clear();

tf\_name.clear();

tf\_age.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_mail.clear();

tf\_card\_no.clear();

tf\_amount.clear();

tf\_last\_date\_used.clear();

tf\_expired\_date.clear();

tf\_pin.clear();

tf\_date\_created.clear();

String query = "SELECT `id` FROM Customer ORDER BY Customer.id DESC LIMIT 1";

String oldid = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

oldid = "" + rs.getString(1);

}

// count +1 new Id

String newId = "" + (Integer.parseInt(oldid) + 1);

tf\_id.setText(newId);

// create today date

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

System.out.println("today is " + todaydate);

tf\_date\_created.setText(todaydate);

tf\_last\_date\_used.setText(todaydate);

// count expired date to 3 years

String[] todaydateAry = todaydate.split("/");

String day = todaydateAry[0];

String month = todaydateAry[1];

String year = todaydateAry[2];

year = "" + (Integer.parseInt(year) + 3);

tf\_expired\_date.setText(day+"/"+month+"/" + year);

}

@FXML

void onAddAction(ActionEvent event) throws SQLException {

if (tf\_id.getText().equals("") || tf\_name.getText().equals("") || tf\_age.getText().equals("")

|| tf\_addr.getText().equals("") || tf\_phone.getText().equals("") || tf\_mail.getText().equals("")

|| tf\_card\_no.getText().equals("") || tf\_amount.getText().equals("") || tf\_pin.getText().equals("")

|| tf\_age.getText().matches(".\*[a-zA-Z]+.\*") || Integer.parseInt(tf\_age.getText()) < 13

|| tf\_phone.getText().matches(".\*[a-zA-Z]+.\*") || tf\_phone.getText().length() < 10

|| !tf\_mail.getText().contains("mail.com") || !tf\_mail.getText().contains("@")

|| tf\_amount.getText().matches(".\*[a-zA-Z]+.\*") || Double.parseDouble(tf\_amount.getText()) < 50000) {

Alert al = new Alert(AlertType.ERROR, "Invaild Input or Data Missing!");

al.showAndWait();

} else {

// create today date

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

// formatter for datepicker

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String age = tf\_age.getText().toString();

String addr = tf\_addr.getText().toString();

String ph = tf\_phone.getText().toString();

String mail = tf\_mail.getText().toString();

String cardno = tf\_card\_no.getText().toString();

String amount = tf\_amount.getText().toString();

// String lastdateused = tf\_last\_date\_used.getText().toString();

String expireddate = tf\_expired\_date.getText().toString();

String pin = tf\_pin.getText().toString();

String datecreated = tf\_date\_created.getText().toString();

System.out.println("expired date is ssssss::::: " + expireddate);

String gender = "";

if (rdo\_male.isSelected()) {

gender = "male";

} else {

gender = "female";

}

try {

String addCustomerQuery = "INSERT INTO `Customer`(`id`, `name`, `age`, `gender`, `address`, `phone`, `email`) "

+ "VALUES ('" + id + "','" + name + "','" + age + "','" + gender + "','" + addr + "','" + ph + "','"

+ mail + "')";

String updateCardQuery = "UPDATE `Card` SET `customerid`='" + id + "',`amount`='" + amount

+ "',`lastuseddate`='" + todaydate + "',`registerdate`='" + datecreated + "',`expireddate`='"

+ expireddate + "',`pin`=" + pin + " WHERE `cardnumber`='" + cardno + "'";

new DBInitialize();

DBInitialize.statement.executeUpdate(addCustomerQuery);

DBInitialize.statement.executeUpdate(updateCardQuery);

// clear

tf\_id.clear();

tf\_name.clear();

tf\_age.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_mail.clear();

tf\_card\_no.clear();

tf\_amount.clear();

tf\_last\_date\_used.clear();

tf\_expired\_date.clear();

tf\_pin.clear();

tf\_date\_created.clear();

// get tabe data

String getTableDataQuery = "SELECT Customer.id, Customer.name, Customer.age, Customer.gender, Customer.address, Customer.phone, Customer.email, Card.cardnumber, Card.amount, Card.lastuseddate, Card.registerdate, Card.expireddate, Card.pin FROM Customer, Card WHERE Customer.id = Card.customerid ORDER BY Customer.id DESC;";

customerData.clear();// clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

CardUser cu = new CardUser();

cu.setCustomrid(getrs.getString(1));

cu.setName(getrs.getString(2));

cu.setAge(getrs.getString(3));

cu.setGender(getrs.getString(4));

cu.setAddress(getrs.getString(5));

cu.setPhone(getrs.getString(6));

cu.setEmail(getrs.getString(7));

cu.setCardno(getrs.getString(8));

cu.setAmount(getrs.getString(9));

cu.setLastdateused(getrs.getString(10));

cu.setRegisterdate(getrs.getString(11));

cu.setExpireddate(getrs.getString(12));

cu.setPin("" + getrs.getInt(13));

customerData.add(cu);

}

// set to table

tb\_customer.refresh();

// tb\_category.setItems(categoryData);

// show alert

Alert al = new Alert(AlertType.INFORMATION, "Item added!");

al.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

} // end of else

}

@FXML

void onUpdateAction(ActionEvent event) throws SQLException {

if (tf\_id.getText().equals("") || tf\_name.getText().equals("") || tf\_age.getText().equals("")

|| tf\_addr.getText().equals("") || tf\_phone.getText().equals("") || tf\_mail.getText().equals("")

|| tf\_card\_no.getText().equals("") || tf\_amount.getText().equals("") || tf\_pin.getText().equals("")

|| tf\_age.getText().matches(".\*[a-zA-Z]+.\*") || Integer.parseInt(tf\_age.getText()) < 13

|| tf\_phone.getText().matches(".\*[a-zA-Z]+.\*") || tf\_phone.getText().length() < 10

|| !tf\_mail.getText().contains("mail.com") || !tf\_mail.getText().contains("@")

|| tf\_amount.getText().matches(".\*[a-zA-Z]+.\*") || Double.parseDouble(tf\_amount.getText()) < 3000) {

Alert al = new Alert(AlertType.ERROR, "Invaild Input or Data Missing!");

al.showAndWait();

} else {

// create today date

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

// formatter for datepicker

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String age = tf\_age.getText().toString();

String addr = tf\_addr.getText().toString();

String ph = tf\_phone.getText().toString();

String mail = tf\_mail.getText().toString();

String cardno = tf\_card\_no.getText().toString();

String amount = tf\_amount.getText().toString();

// String lastdateused = tf\_last\_date\_used.getText().toString();

String expireddate = tf\_expired\_date.getText().toString();

String pin = tf\_pin.getText().toString();

String datecreated = tf\_date\_created.getText().toString();

String gender = "";

if (rdo\_male.isSelected()) {

gender = "male";

} else {

gender = "female";

}

String updateCustomerQuery = "UPDATE `Customer` SET `name`='" + name + "',`age`='" + age + "',`gender`='"

+ gender + "',`address`='" + addr + "',`phone`='" + ph + "',`email`='" + mail + "' WHERE `id` = '"

+ id + "'";

String updateCardQuery = "UPDATE `Card` SET `customerid`='" + id + "',`amount`='" + amount

+ "',`lastuseddate`='" + todaydate + "',`registerdate`='" + datecreated + "',`expireddate`='"

+ expireddate + "',`pin`=" + pin + " WHERE `cardnumber`='" + cardno + "'";

new DBInitialize();

DBInitialize.statement.executeUpdate(updateCustomerQuery);

DBInitialize.statement.executeUpdate(updateCardQuery);

// clear

tf\_id.clear();

tf\_name.clear();

tf\_age.clear();

tf\_addr.clear();

tf\_phone.clear();

tf\_mail.clear();

tf\_card\_no.clear();

tf\_amount.clear();

tf\_last\_date\_used.clear();

tf\_expired\_date.clear();

tf\_pin.clear();

tf\_date\_created.clear();

// get tabe data

String getTableDataQuery = "SELECT Customer.id, Customer.name, Customer.age, Customer.gender, Customer.address, Customer.phone, Customer.email, Card.cardnumber, Card.amount, Card.lastuseddate, Card.registerdate, Card.expireddate, Card.pin FROM Customer, Card WHERE Customer.id = Card.customerid ORDER BY Customer.id DESC;";

customerData.clear();// clear category data

new DBInitialize();

ResultSet getrs = DBInitialize.statement.executeQuery(getTableDataQuery);

while (getrs.next()) {

CardUser cu = new CardUser();

cu.setCustomrid(getrs.getString(1));

cu.setName(getrs.getString(2));

cu.setAge(getrs.getString(3));

cu.setGender(getrs.getString(4));

cu.setAddress(getrs.getString(5));

cu.setPhone(getrs.getString(6));

cu.setEmail(getrs.getString(7));

cu.setCardno(getrs.getString(8));

cu.setAmount(getrs.getString(9));

cu.setLastdateused(getrs.getString(10));

cu.setRegisterdate(getrs.getString(11));

cu.setExpireddate(getrs.getString(12));

cu.setPin("" + getrs.getInt(13));

customerData.add(cu);

}

// set to table

tb\_customer.refresh();

// show alert

Alert al = new Alert(AlertType.INFORMATION, "Item updated!");

al.showAndWait();

}

}// end of first else

}

**AdminPopularItemController.java**

package controller;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ResourceBundle;

import database.DBInitialize;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import model.ProductItem;

public class AdminPopularItemController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private TableView<ProductItem> tb\_popular;

private TableColumn<ProductItem, String> col\_item\_barcode;

private TableColumn<ProductItem, String> col\_item\_name;

private TableColumn<ProductItem, String> col\_item\_categroy;

private TableColumn<ProductItem, String> col\_item\_price;

private TableColumn<ProductItem, String> col\_item\_supplier;

private TableColumn<ProductItem, String> col\_item\_dateadded;

private TableColumn<ProductItem, String> col\_item\_stock;

private TableColumn<ProductItem, String> col\_item\_expire\_date;

private TableColumn<ProductItem, String> col\_item\_count;

private ObservableList<ProductItem> popularData = FXCollections.observableArrayList();

@FXML

void initialize() throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

assert tb\_popular != null : "fx:id=\"tb\_popular\" was not injected: check your FXML file 'Admin\_popular\_item.fxml'.";

col\_item\_barcode = new TableColumn<ProductItem, String>("Barcode");

col\_item\_name = new TableColumn<ProductItem, String>("Name");

col\_item\_categroy = new TableColumn<ProductItem, String>("Category");

col\_item\_price = new TableColumn<ProductItem, String>("Price");

col\_item\_supplier = new TableColumn<ProductItem, String>("Supplier");

col\_item\_dateadded = new TableColumn<ProductItem, String>("Date Added");

col\_item\_stock = new TableColumn<ProductItem, String>("Stock");

col\_item\_expire\_date = new TableColumn<ProductItem, String>("Expired Date");

col\_item\_count = new TableColumn<ProductItem, String>("Count");

col\_item\_barcode.setMinWidth(190.0);

col\_item\_name.setMinWidth(200.0);

col\_item\_categroy.setMinWidth(160.0);

col\_item\_price.setMinWidth(90.0);

col\_item\_supplier.setMinWidth(170.0);

col\_item\_dateadded.setMinWidth(120.0);

col\_item\_stock.setMinWidth(90.0);

col\_item\_expire\_date.setMinWidth(140.0);

col\_item\_count.setMinWidth(60.0);

col\_item\_barcode.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_categroy.setStyle("-fx-font-size: 18");

col\_item\_price.setStyle("-fx-font-size: 18");

col\_item\_supplier.setStyle("-fx-font-size: 18");

col\_item\_dateadded.setStyle("-fx-font-size: 18");

col\_item\_stock.setStyle("-fx-font-size: 18");

col\_item\_expire\_date.setStyle("-fx-font-size: 18");

col\_item\_count.setStyle("-fx-font-size: 18");

col\_item\_barcode.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("barcode"));

col\_item\_name.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("name"));

col\_item\_categroy.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("categoryname"));

col\_item\_price.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("price"));

col\_item\_supplier.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("suppliername"));

col\_item\_dateadded.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("dateadded"));

col\_item\_stock.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("stockamount"));

col\_item\_expire\_date.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("expiredate"));

col\_item\_count.setCellValueFactory(

new PropertyValueFactory<ProductItem, String>("count"));

tb\_popular.getColumns().addAll(col\_item\_barcode, col\_item\_name, col\_item\_categroy, col\_item\_price, col\_item\_supplier, col\_item\_dateadded, col\_item\_stock, col\_item\_expire\_date, col\_item\_count);

//get table data

new DBInitialize().DBInitialize();

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate, productitems.count FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.count DESC LIMIT 25";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while(rs.next()) {

ProductItem product = new ProductItem();

product.setBarcode(rs.getString(1));

product.setName(rs.getString(2));

product.setCategoryname(rs.getString(3));

product.setPrice(rs.getString(4));

product.setSuppliername(rs.getString(5));

product.setDateadded(rs.getString(6));

product.setStockamount(rs.getString(7));

product.setExpiredate(rs.getString(8));

product.setCount(rs.getInt(9));

popularData.add(product);

}

tb\_popular.setItems(popularData);

}

}

**AdminProductController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXComboBox;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.MenuItem;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.KeyEvent;

import model.ProductItem;

public class AdminProductController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXTextField tf\_barcode;

@FXML

private JFXTextField tf\_name;

@FXML

private JFXComboBox<String> cbo\_category;

@FXML

private JFXComboBox<String> cbo\_supplier;

@FXML

private JFXTextField tf\_price;

@FXML

private JFXTextField tf\_date\_added;

@FXML

private JFXTextField tf\_stock;

@FXML

private JFXTextField tf\_expired\_date;

@FXML

private JFXButton bt\_new;

@FXML

private JFXTextField tf\_name\_search;

@FXML

private JFXTextField tf\_barcode\_search;

@FXML

private JFXButton bt\_add;

@FXML

private JFXButton bt\_update;

@FXML

private TableView<ProductItem> tb\_product\_item;

private TableColumn<ProductItem, String> col\_item\_barcode;

private TableColumn<ProductItem, String> col\_item\_name;

private TableColumn<ProductItem, String> col\_item\_categroy;

private TableColumn<ProductItem, String> col\_item\_price;

private TableColumn<ProductItem, String> col\_item\_supplier;

private TableColumn<ProductItem, String> col\_item\_dateadded;

private TableColumn<ProductItem, String> col\_item\_stock;

private TableColumn<ProductItem, String> col\_item\_expire\_date;

private ObservableList<ProductItem> productData = FXCollections.observableArrayList();

private ObservableList<String> categoryData = FXCollections.observableArrayList();

private ObservableList<String> supplierData = FXCollections.observableArrayList();

private int index = 0;

private int indexsupplier = 0;

private ResultSet rs;

private ResultSet rsCategory;

private ResultSet rsSupplier;

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tb\_product\_item != null : "fx:id=\"tb\_product\_item\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_barcode != null : "fx:id=\"tf\_barcode\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_price != null : "fx:id=\"tf\_price\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_date\_added != null : "fx:id=\"tf\_date\_added\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_stock != null : "fx:id=\"tf\_stock\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_expired\_date != null : "fx:id=\"tf\_expired\_date\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert bt\_add != null : "fx:id=\"bt\_add\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert bt\_update != null : "fx:id=\"bt\_update\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert cbo\_category != null : "fx:id=\"cbo\_category\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert cbo\_supplier != null : "fx:id=\"cbo\_supplier\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_name\_search != null : "fx:id=\"tf\_name\_search\" was not injected: check your FXML file 'admin\_product.fxml'.";

assert tf\_barcode\_search != null : "fx:id=\"tf\_barcode\_search\" was not injected: check your FXML file 'admin\_product.fxml'.";

col\_item\_barcode = new TableColumn<ProductItem, String>("Barcode");

col\_item\_name = new TableColumn<ProductItem, String>("Name");

col\_item\_categroy = new TableColumn<ProductItem, String>("Category");

col\_item\_price = new TableColumn<ProductItem, String>("Price");

col\_item\_supplier = new TableColumn<ProductItem, String>("Supplier");

col\_item\_dateadded = new TableColumn<ProductItem, String>("Date Added");

col\_item\_stock = new TableColumn<ProductItem, String>("Stock");

col\_item\_expire\_date = new TableColumn<ProductItem, String>("Expired Date");

col\_item\_barcode.setMinWidth(190.0);

col\_item\_name.setMinWidth(200.0);

col\_item\_categroy.setMinWidth(160.0);

col\_item\_price.setMinWidth(90.0);

col\_item\_supplier.setMinWidth(170.0);

col\_item\_dateadded.setMinWidth(120.0);

col\_item\_stock.setMinWidth(90.0);

col\_item\_expire\_date.setMinWidth(140.0);

col\_item\_barcode.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_categroy.setStyle("-fx-font-size: 18");

col\_item\_price.setStyle("-fx-font-size: 18");

col\_item\_supplier.setStyle("-fx-font-size: 18");

col\_item\_dateadded.setStyle("-fx-font-size: 18");

col\_item\_stock.setStyle("-fx-font-size: 18");

col\_item\_expire\_date.setStyle("-fx-font-size: 18");

col\_item\_barcode.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("barcode"));

col\_item\_name.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("name"));

col\_item\_categroy.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("categoryname"));

col\_item\_price.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("price"));

col\_item\_supplier.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("suppliername"));

col\_item\_dateadded.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("dateadded"));

col\_item\_stock.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("stockamount"));

col\_item\_expire\_date.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("expiredate"));

tb\_product\_item.getColumns().addAll(col\_item\_barcode, col\_item\_name, col\_item\_categroy, col\_item\_price,

col\_item\_supplier, col\_item\_dateadded, col\_item\_stock, col\_item\_expire\_date);

// get category and set to conmbobox

String categoryquery = "SELECT \* FROM `productcategory`;";

new DBInitialize();

rsCategory = DBInitialize.statement.executeQuery(categoryquery);

while (rsCategory.next()) {

String categoryname = rsCategory.getString(2);

categoryData.add(categoryname);

}

cbo\_category.setItems(categoryData);

cbo\_category.setValue("Snacks");

// get supplier name and set to combobox

String supplierquery = "SELECT \* FROM `supplier`;";

new DBInitialize();

rsSupplier = DBInitialize.statement.executeQuery(supplierquery);

while (rsSupplier.next()) {

String suppliername = rsSupplier.getString(2);

supplierData.add(suppliername);

}

cbo\_supplier.setItems(supplierData);

cbo\_supplier.setValue(supplierData.get(1));

new DBInitialize().DBInitialize();

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC;";

new DBInitialize();

rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

ProductItem product = new ProductItem();

product.setBarcode(rs.getString(1));

product.setName(rs.getString(2));

product.setCategoryname(rs.getString(3));

product.setPrice(rs.getString(4));

product.setSuppliername(rs.getString(5));

product.setDateadded(rs.getString(6));

product.setStockamount(rs.getString(7));

product.setExpiredate(rs.getString(8));

productData.add(product);

}

tb\_product\_item.setItems(productData);

tb\_product\_item.setRowFactory(t -> {

TableRow<ProductItem> row = new TableRow<>();

row.setOnMouseClicked(e -> {

// get data from selected row

if (e.getClickCount() == 2 && (!row.isEmpty())) {

ProductItem product = tb\_product\_item.getSelectionModel().getSelectedItem();

System.out.println("Double click is: " + product.getName());

// get category name

String categoryfromproducttable = product.getCategoryname();

for (int i = 0; i < categoryData.size(); i++) {

if (categoryfromproducttable.equals(categoryData.get(i))) {

index = i;

}

}

// get supplier name

String supplierfromproducttable = product.getSuppliername();

for (int i = 0; i < supplierData.size(); i++) {

if (supplierfromproducttable.equals(supplierData.get(i))) {

indexsupplier = i;

}

}

// add data to text fields

tf\_barcode.setText(product.getBarcode());

tf\_name.setText(product.getName());

cbo\_category.setValue(categoryData.get(index));

tf\_price.setText(product.getPrice());

cbo\_supplier.setValue(supplierData.get(indexsupplier));

tf\_date\_added.setText(product.getDateadded());

tf\_stock.setText(product.getStockamount());

tf\_expired\_date.setText(product.getExpiredate());

}

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

ProductItem p = tb\_product\_item.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION, "Are U Sure To Delete " + p.getName() + " ?",

ButtonType.YES, ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

// do stuff

String removequery = "DELETE FROM `productitems` WHERE productitems.barcode = '"

+ p.getBarcode() + "';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(removequery);

// update table

// update table data

new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC;";

new DBInitialize();

ResultSet rss = DBInitialize.statement.executeQuery(queryupdatetable);

productData.clear();

while (rss.next()) {

ProductItem product = new ProductItem();

product.setBarcode(rss.getString(1));

product.setName(rss.getString(2));

product.setCategoryname(rss.getString(3));

product.setPrice(rss.getString(4));

product.setSuppliername(rss.getString(5));

product.setDateadded(rss.getString(6));

product.setStockamount(rss.getString(7));

product.setExpiredate(rss.getString(8));

productData.add(product);

}

// tb\_product\_item.getItems().clear();

tb\_product\_item.refresh();

/\*

\* //show alert Alert al = new Alert(AlertType.INFORMATION, "Item removed!");

\* al.showAndWait();

\*/

} catch (ClassNotFoundException | SQLException | InstantiationException

| IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

});

rowMenu.getItems().addAll(removeItem);

// only display context menu for non-null items:

row.contextMenuProperty().bind(Bindings.when(Bindings.isNotNull(row.itemProperty())).then(rowMenu)

.otherwise((ContextMenu) null));

});

return row;

});

}

@FXML

void onBtUpdateAction(ActionEvent event)

throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

if (tf\_barcode.getText().toString().equals("") || tf\_name.getText().toString().equals("")

|| tf\_price.getText().toString().equals("") || tf\_stock.getText().toString().equals("")

|| tf\_expired\_date.getText().toString().equals("") || tf\_price.getText().matches(".\*[a-zA-Z]+.\*")

|| Double.parseDouble(tf\_price.getText().toString()) <= 0 || tf\_stock.getText().matches(".\*[a-zA-Z]+.\*")

|| Double.parseDouble(tf\_stock.getText().toString()) <= 0 || tf\_expired\_date.getText().length() < 9) {

Alert al = new Alert(AlertType.ERROR, "Input Error or Data Missing!");

al.showAndWait();

} else {

String name = tf\_name.getText().toString();

String categoryname = cbo\_category.getValue().toString();

String dateadded = tf\_date\_added.getText().toString();

String expireddate = tf\_expired\_date.getText().toString();

String price = tf\_price.getText().toString();

String barcode = tf\_barcode.getText().toString();

String suppliername = cbo\_supplier.getValue().toString();

String stockamount = tf\_stock.getText().toString();

// get category id by name

String categoryQuery = "SELECT `id` FROM `productcategory` WHERE productcategory.name = '" + categoryname

+ "'";

int categoryid = 0;

// new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(categoryQuery);

if (rs.next()) {

categoryid = rs.getInt(1);

} else {

// show alert

Alert alert = new Alert(AlertType.ERROR, "Try Again1");

alert.showAndWait();

}

// get category id by name

String supplierQuery = "SELECT `id` FROM `supplier` WHERE supplier.companyname = '" + suppliername + "'";

System.out.println("Supplier name is : " + suppliername);

int supplierid = 0;

// new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet r = DBInitialize.statement.executeQuery(supplierQuery);

if (r.next()) {

supplierid = r.getInt(1);

System.out.println("Supplier id is : " + supplierid);

} else {

// show alert

Alert alert = new Alert(AlertType.ERROR, "Try Again2");

alert.showAndWait();

}

String query = "UPDATE `productitems` SET `name`= '" + name + "',`categoryid`=" + categoryid

+ ",`dateadded`= '" + dateadded + "' ,`expireddate`= '" + expireddate + "' ,`price`= '" + price

+ "' ,`supplierid`=" + supplierid + " ,`stockamount`= '" + stockamount + "' WHERE `barcode`= '"

+ barcode + "'";

// new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(query);

// clear

tf\_name.clear();

tf\_date\_added.clear();

tf\_expired\_date.clear();

tf\_price.clear();

tf\_barcode.clear();

tf\_stock.clear();

// update table data

// new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC;";

new DBInitialize();

ResultSet rss = DBInitialize.statement.executeQuery(queryupdatetable);

productData.clear();

while (rss.next()) {

ProductItem product = new ProductItem();

product.setBarcode(rss.getString(1));

product.setName(rss.getString(2));

product.setCategoryname(rss.getString(3));

product.setPrice(rss.getString(4));

product.setSuppliername(rss.getString(5));

product.setDateadded(rss.getString(6));

product.setStockamount(rss.getString(7));

product.setExpiredate(rss.getString(8));

productData.add(product);

}

// tb\_product\_item.getItems().clear();

tb\_product\_item.refresh();

// tb\_product\_item.setItems(productData);

// show alert

Alert alert = new Alert(AlertType.INFORMATION, "Success! Item is update to database.");

alert.showAndWait();

} // end of else

}

// ---------------------------------------------------------------------------------------------------------------------==

@FXML

void onbtAddAction(ActionEvent event)

throws SQLException, ClassNotFoundException, InstantiationException, IllegalAccessException {

if (tf\_barcode.getText().toString().equals("") || tf\_name.getText().toString().equals("")

|| tf\_price.getText().toString().equals("") || tf\_stock.getText().toString().equals("")

|| tf\_expired\_date.getText().toString().equals("") || tf\_price.getText().matches(".\*[a-zA-Z]+.\*")

|| Double.parseDouble(tf\_price.getText().toString()) <= 0 || tf\_stock.getText().matches(".\*[a-zA-Z]+.\*")

|| Double.parseDouble(tf\_stock.getText().toString()) <= 0 || tf\_expired\_date.getText().length() < 9) {

Alert al = new Alert(AlertType.ERROR, "Input Error or Data Missing!");

al.showAndWait();

} else {

String name = tf\_name.getText().toString();

String categoryname = cbo\_category.getValue().toString();

String dateadded = tf\_date\_added.getText().toString();

String expireddate = tf\_expired\_date.getText().toString();

String price = tf\_price.getText().toString();

String barcode = tf\_barcode.getText().toString();

String suppliername = cbo\_supplier.getValue().toString();

String stockamount = tf\_stock.getText().toString();

try {

// get category id by name

String categoryQuery = "SELECT `id` FROM `productcategory` WHERE productcategory.name = '" + categoryname

+ "'";

int categoryid = 0;

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(categoryQuery);

if (rs.next()) {

categoryid = rs.getInt(1);

} else {

// show alert

Alert alert = new Alert(AlertType.ERROR, "Try Again1");

alert.showAndWait();

}

// get category id by name

String supplierQuery = "SELECT `id` FROM `supplier` WHERE supplier.companyname = '" + suppliername + "'";

System.out.println("Supplier name is : " + suppliername);

int supplierid = 0;

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet r = DBInitialize.statement.executeQuery(supplierQuery);

if (r.next()) {

supplierid = r.getInt(1);

System.out.println("Supplier id is : " + supplierid);

} else {

// show alert

Alert alert = new Alert(AlertType.ERROR, "Try Again2");

alert.showAndWait();

}

String query = "INSERT INTO `productitems`(`name`, `categoryid`, `dateadded`, `expireddate`, `price`, `barcode`, `supplierid`, `stockamount`, `count`) VALUES"

+ " ('" + name + "'," + categoryid + ",'" + dateadded + "','" + expireddate + "','" + price + "','"

+ barcode + "'," + supplierid + ",'" + stockamount + "', 0);";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(query);

// update supplier table lastsupplied date

String pat = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pat).format(new Date());

System.out.println("today is " + todaydate);

String upquery = "UPDATE `supplier` SET `lastdatesupplied`='" + todaydate + "' WHERE `id`='" + supplierid

+ "'";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(upquery);

// clear

tf\_name.clear();

tf\_date\_added.clear();

tf\_expired\_date.clear();

tf\_price.clear();

tf\_barcode.clear();

tf\_stock.clear();

// update table data

new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC ;";

new DBInitialize();

ResultSet rss = DBInitialize.statement.executeQuery(queryupdatetable);

productData.clear();

while (rss.next()) {

ProductItem product = new ProductItem();

product.setBarcode(rss.getString(1));

product.setName(rss.getString(2));

product.setCategoryname(rss.getString(3));

product.setPrice(rss.getString(4));

product.setSuppliername(rss.getString(5));

product.setDateadded(rss.getString(6));

product.setStockamount(rss.getString(7));

product.setExpiredate(rss.getString(8));

productData.add(product);

}

// tb\_product\_item.getItems().clear();

tb\_product\_item.refresh();

// tb\_product\_item.setItems(productData);

// show alert

Alert alert = new Alert(AlertType.INFORMATION, "Success! One product items is added to database.");

alert.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

} // end of else

}

@FXML

void onbtNewAction(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// clear

tf\_name.clear();

tf\_date\_added.clear();

tf\_expired\_date.clear();

tf\_price.clear();

tf\_barcode.clear();

tf\_stock.clear();

String query = "SELECT `barcode` FROM `productitems` ORDER BY productitems.barcode DESC LIMIT 1";

String oldbarcode = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

oldbarcode = "" + rs.getString(1);

}

// count +1 new Id

String newbarcode = "" + (Long.parseLong(oldbarcode) + 1);

tf\_barcode.setText(newbarcode);

// create today date

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

System.out.println("today is " + todaydate);

tf\_date\_added.setText(todaydate);

}

///////////////////////

// barcode search

@FXML

void onBarcodeSearchActionn(KeyEvent event) {

String searchKey = tf\_barcode\_search.getText().toString();

System.out.println("key entered is : " + searchKey);

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id AND productitems.barcode LIKE '"

+ searchKey + "%'";

// new DBInitialize().DBInitialize();

System.out.println("working");

try {

// ResultSet rs = st.executeQuery("SELECT \* FROM USER");

ResultSet rs = DBInitialize.statement.executeQuery(query);

ObservableList<ProductItem> row = FXCollections.observableArrayList();

while (rs.next()) {

ProductItem p = new ProductItem();

p.setBarcode(rs.getString(1));

p.setName(rs.getString(2));

p.setCategoryname(rs.getString(3));

p.setPrice(rs.getString(4));

p.setSuppliername(rs.getString(5));

p.setDateadded(rs.getString(6));

p.setStockamount(rs.getString(7));

p.setExpiredate(rs.getString(8));

row.add(p);

}

tb\_product\_item.setItems(row);

// System.out.println("working1"+data);

// tb\_total\_item.getItems().clear();

// tb\_total\_item.setItems(data);

System.out.println("working2");

// data.getItems().addAll(row);

} catch (SQLException ex) {

}

}

///////////////

// name search

@FXML

void onNameSearchAction(KeyEvent event) {

String searchKey = tf\_name\_search.getText().toString();

System.out.println("key entered is : " + searchKey);

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id AND productitems.name LIKE '"

+ searchKey + "%'";

// new DBInitialize().DBInitialize();

System.out.println("working");

try {

// ResultSet rs = st.executeQuery("SELECT \* FROM USER");

ResultSet rs = DBInitialize.statement.executeQuery(query);

ObservableList<ProductItem> row = FXCollections.observableArrayList();

while (rs.next()) {

ProductItem p = new ProductItem();

p.setBarcode(rs.getString(1));

p.setName(rs.getString(2));

p.setCategoryname(rs.getString(3));

p.setPrice(rs.getString(4));

p.setSuppliername(rs.getString(5));

p.setDateadded(rs.getString(6));

p.setStockamount(rs.getString(7));

p.setExpiredate(rs.getString(8));

row.add(p);

}

tb\_product\_item.setItems(row);

System.out.println("working2");

// data.getItems().addAll(row);

} catch (SQLException ex) {

}

}

}

**AdminPromotionController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.MenuItem;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import model.Promotion;

public class AdminPromotionController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private TableView<Promotion> tb\_promo;

private TableColumn<Promotion, String> col\_item\_id;

private TableColumn<Promotion, String> col\_item\_name;

private TableColumn<Promotion, String> col\_item\_product\_id;

private TableColumn<Promotion, String> col\_item\_percentage;

private TableColumn<Promotion, String> col\_item\_more;

private TableColumn<Promotion, String> col\_item\_product\_name;

private ObservableList<Promotion> promoData = FXCollections.observableArrayList();

@FXML

private JFXTextField tf\_id;

@FXML

private JFXTextField tf\_name;

@FXML

private JFXTextField tf\_product\_id;

@FXML

private JFXTextField tf\_percentage;

@FXML

private JFXTextField tf\_buy;

@FXML

private JFXTextField tf\_get;

@FXML

private JFXButton bt\_add;

@FXML

private JFXTextField tf\_product\_name;

@FXML

private JFXButton bt\_new;

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tb\_promo != null : "fx:id=\"tb\_promo\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_id != null : "fx:id=\"tf\_id\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_product\_id != null : "fx:id=\"tf\_product\_id\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_percentage != null : "fx:id=\"tf\_percentage\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_buy != null : "fx:id=\"tf\_buy\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_get != null : "fx:id=\"tf\_get\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert bt\_add != null : "fx:id=\"bt\_add\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert tf\_product\_name != null : "fx:id=\"tf\_product\_name\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Admin\_promotion.fxml'.";

col\_item\_id = new TableColumn<Promotion, String>("ID");

col\_item\_name = new TableColumn<Promotion, String>("Name");

col\_item\_product\_id = new TableColumn<Promotion, String>("Product ID");

col\_item\_product\_name = new TableColumn<Promotion, String>("Product Name");

col\_item\_percentage = new TableColumn<Promotion, String>("Percentage");

col\_item\_more = new TableColumn<Promotion, String>("More");

col\_item\_id.setMinWidth(100.0);

col\_item\_name.setMinWidth(200.0);

col\_item\_product\_id.setMinWidth(200.0);

col\_item\_product\_name.setMinWidth(200.0);

col\_item\_percentage.setMinWidth(120.0);

col\_item\_more.setMinWidth(150.0);

col\_item\_id.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_product\_id.setStyle("-fx-font-size: 18");

col\_item\_product\_name.setStyle("-fx-font-size: 18");

col\_item\_percentage.setStyle("-fx-font-size: 18");

col\_item\_more.setStyle("-fx-font-size: 18");

col\_item\_id.setCellValueFactory(new PropertyValueFactory<Promotion, String>("id"));

col\_item\_name.setCellValueFactory(new PropertyValueFactory<Promotion, String>("name"));

col\_item\_product\_id.setCellValueFactory(new PropertyValueFactory<Promotion, String>("productId"));

col\_item\_product\_name.setCellValueFactory(new PropertyValueFactory<Promotion, String>("productName"));

col\_item\_percentage.setCellValueFactory(new PropertyValueFactory<Promotion, String>("percentage"));

col\_item\_more.setCellValueFactory(new PropertyValueFactory<Promotion, String>("more"));

tb\_promo.getColumns().addAll(col\_item\_id, col\_item\_name, col\_item\_product\_id, col\_item\_product\_name,

col\_item\_percentage, col\_item\_more);

// get data from db

String query = "SELECT promotion.id, promotion.name, promotion.productid, productitems.name,promotion.percentage, promotion.description FROM promotion, productitems WHERE promotion.productid = productitems.barcode";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

Promotion pro = new Promotion();

pro.setId(rs.getString(1));

pro.setName(rs.getString(2));

pro.setProductId(rs.getString(3));

pro.setProductName(rs.getString(4));

pro.setPercentage(rs.getString(5));

pro.setMore(rs.getString(6));

promoData.add(pro);

}

// set data to table

tb\_promo.setItems(promoData);

// row action

tb\_promo.setRowFactory(t -> {

TableRow<Promotion> row = new TableRow<>();

row.setOnMouseClicked(e -> {

// get data from selected row

if (e.getClickCount() == 2 && (!row.isEmpty())) {

Promotion pro = tb\_promo.getSelectionModel().getSelectedItem();

System.out.println("Double click is: " + pro.getName());

// string processing / cut buy and get

String more = pro.getMore();

String[] buy = more.split(" ");

// String get = more;

// set data to tf

tf\_id.setText(pro.getId());

tf\_name.setText(pro.getName());

tf\_product\_id.setText(pro.getProductId());

tf\_product\_name.setText(pro.getProductName());

;

tf\_percentage.setText(pro.getPercentage());

tf\_buy.setText(buy[1]);

tf\_get.setText(buy[3]);

}

});

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Promotion pro = tb\_promo.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION, "Are U Sure To Delete " + pro.getName() + " ?",

ButtonType.YES, ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

// do stuff

String removequery = "DELETE FROM `promotion` WHERE promotion.id = '" + pro.getId() + "';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(removequery);

// update table

// update table data

new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT promotion.id, promotion.name, promotion.productid, productitems.name,promotion.percentage, promotion.description FROM promotion, productitems WHERE promotion.productid = productitems.barcode";

new DBInitialize();

ResultSet rsu = DBInitialize.statement.executeQuery(queryupdatetable);

promoData.clear();

while (rsu.next()) {

Promotion p = new Promotion();

p.setId(rsu.getString(1));

p.setName(rsu.getString(2));

p.setProductId(rsu.getString(3));

p.setProductName(rsu.getString(4));

p.setPercentage(rsu.getString(5));

p.setMore(rsu.getString(6));

promoData.add(p);

}

// tb\_product\_item.getItems().clear();

tb\_promo.refresh();

/\*

\* //show alert Alert al = new Alert(AlertType.INFORMATION, "Item removed!");

\* al.showAndWait();

\*/

// show alert

Alert al = new Alert(AlertType.INFORMATION, "Item deleted!");

al.showAndWait();

} catch (ClassNotFoundException | SQLException | InstantiationException

| IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

});

rowMenu.getItems().addAll(removeItem);

// only display context menu for non-null items:

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty())).then(rowMenu).otherwise((ContextMenu) null));

return row;

});

}

@FXML

void onProductIDSearch(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

if(tf\_product\_id.getText().equals("")) {

Alert al = new Alert(AlertType.ERROR, "Please enter product barcode!");

al.showAndWait();

}else {

String productbarcode = tf\_product\_id.getText().toString();

String searchPQuery = "SELECT productitems.name FROM `productitems` WHERE productitems.barcode = '"

+ productbarcode + "'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsp = DBInitialize.statement.executeQuery(searchPQuery);

String productname = "";

if (rsp.next()) {

productname = rsp.getString(1);

}

else {

Alert al = new Alert(AlertType.ERROR, "Invaild Product Barcode!");

al.showAndWait();

}

tf\_product\_name.setText(productname);

}//end of else

}

@FXML

void onNewAction(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

tf\_id.clear();

tf\_name.clear();

tf\_product\_id.clear();

tf\_product\_name.clear();

tf\_percentage.clear();

tf\_buy.clear();

tf\_get.clear();

String query = "SELECT promotion.id FROM promotion ORDER BY promotion.id DESC LIMIT 1";

String oldid = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

oldid = "" + rs.getInt(1);

}

// count +1 new Id

String newId = "" + (Integer.parseInt(oldid) + 1);

tf\_id.setText(newId);

}

@FXML

void onAddAction(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

if(tf\_id.getText().equals("") ||tf\_name.getText().equals("") || tf\_product\_id.getText().equals("") ||

tf\_percentage.getText().matches(".\*[a-zA-Z]+.\*")

) {

Alert al = new Alert(AlertType.ERROR, "Invalid input or data missing!");

al.showAndWait();

}

else {

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String productid = tf\_product\_id.getText().toString();

String percentage = tf\_percentage.getText().toString();

String buy = tf\_buy.getText().toString();

String get = tf\_get.getText().toString();

if (percentage.equals("")) {

percentage = "0";

}

if (buy.equals("")) {

buy = "0";

}

if (get.equals("")) {

get = "0";

}

String desc = "Buy " + buy + " Get " + get;

try {

String addquery = "INSERT INTO `promotion`(`id`, `name`, `productid`, `percentage`, `description`) "

+ "VALUES ('" + id + "','" + name + "','" + productid + "','" + percentage + "','" + desc + "')";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(addquery);

// update table

String query = "SELECT promotion.id, promotion.name, promotion.productid, productitems.name,promotion.percentage, promotion.description FROM promotion, productitems WHERE promotion.productid = productitems.barcode";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

// clear promodata array

promoData.clear();

while (rs.next()) {

Promotion pro = new Promotion();

pro.setId(rs.getString(1));

pro.setName(rs.getString(2));

pro.setProductId(rs.getString(3));

pro.setProductName(rs.getString(4));

pro.setPercentage(rs.getString(5));

pro.setMore(rs.getString(6));

promoData.add(pro);

}

// set data to table

tb\_promo.refresh();

tf\_id.clear();

tf\_name.clear();

tf\_product\_id.clear();

tf\_product\_name.clear();

tf\_percentage.clear();

tf\_buy.clear();

tf\_get.clear();

// show alert

Alert al = new Alert(AlertType.INFORMATION, "Item added!");

al.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}//end of else

}

@FXML

void onUpdateAction(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

if(tf\_id.getText().equals("") ||tf\_name.getText().equals("") || tf\_product\_id.getText().equals("") ||

tf\_percentage.getText().matches(".\*[a-zA-Z]+.\*")

) {

Alert al = new Alert(AlertType.ERROR, "Invalid input or data missing!");

al.showAndWait();

}

else {

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String productid = tf\_product\_id.getText().toString();

String percentage = tf\_percentage.getText().toString();

String buy = tf\_buy.getText().toString();

String get = tf\_get.getText().toString();

String desc = "Buy " + buy + " Get " + get;

String updatequery = "UPDATE `promotion` SET `name`='" + name + "',`productid`='" + productid

+ "',`percentage`='" + percentage + "',`description`='" + desc + "' WHERE promotion.id = '" + id + "';";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(updatequery);

// update table

String query = "SELECT promotion.id, promotion.name, promotion.productid, productitems.name,promotion.percentage, promotion.description FROM promotion, productitems WHERE promotion.productid = productitems.barcode";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

// clear promodata array

promoData.clear();

while (rs.next()) {

Promotion pro = new Promotion();

pro.setId(rs.getString(1));

pro.setName(rs.getString(2));

pro.setProductId(rs.getString(3));

pro.setProductName(rs.getString(4));

pro.setPercentage(rs.getString(5));

pro.setMore(rs.getString(6));

promoData.add(pro);

}

// set data to table

tb\_promo.refresh();

tf\_id.clear();

tf\_name.clear();

tf\_product\_id.clear();

tf\_product\_name.clear();

tf\_percentage.clear();

tf\_buy.clear();

tf\_get.clear();

// show alert

Alert al = new Alert(AlertType.INFORMATION, "Item updated!");

al.showAndWait();

}

}//end of else

}

**AdminSupplierController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.MenuItem;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import model.Supplier;

public class AdminSupplierController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXTextField tf\_id;

@FXML

private JFXButton bt\_add;

@FXML

private JFXButton bt\_update;

@FXML

private JFXButton bt\_new;

@FXML

private TableView<Supplier> tb\_supplier;

private TableColumn<Supplier, String> col\_item\_id;

private TableColumn<Supplier, String> col\_item\_name;

private TableColumn<Supplier, String> col\_item\_supplied\_last;

private ObservableList<Supplier> supplierData = FXCollections.observableArrayList();

@FXML

private JFXTextField tf\_name;

@FXML

private JFXTextField tf\_supplied\_date;

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tb\_supplier != null : "fx:id=\"tb\_supplier\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

assert bt\_add != null : "fx:id=\"bt\_add\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

assert tf\_id != null : "fx:id=\"tf\_id\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

assert tb\_supplier != null : "fx:id=\"tb\_supplier\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

assert tf\_supplied\_date != null : "fx:id=\"tf\_supplied\_date\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Admin\_supplier.fxml'.";

tf\_id.setEditable(false);

//tf\_supplied\_date.setEditable(false);

col\_item\_id = new TableColumn<Supplier, String>("ID");

col\_item\_name = new TableColumn<Supplier, String>("Compant Name");

col\_item\_supplied\_last = new TableColumn<Supplier, String>("Last Supplied Date");

col\_item\_id.setMinWidth(110.0);

col\_item\_name.setMinWidth(380.0);

col\_item\_supplied\_last.setMinWidth(180.0);

col\_item\_id.setStyle("-fx-font-size: 18");

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_supplied\_last.setStyle("-fx-font-size: 18");

col\_item\_id.setCellValueFactory(new PropertyValueFactory<Supplier, String>("id"));

col\_item\_name.setCellValueFactory(new PropertyValueFactory<Supplier, String>("name"));

col\_item\_supplied\_last.setCellValueFactory(new PropertyValueFactory<Supplier, String>("lastSupplied"));

tb\_supplier.getColumns().addAll(col\_item\_id, col\_item\_name, col\_item\_supplied\_last);

// get data from db

String query = "SELECT \* FROM `supplier`;";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

Supplier su = new Supplier();

su.setId(rs.getString(1));

su.setName(rs.getString(2));

su.setLastSupplied(rs.getString(3));

supplierData.add(su);

}

// set data to table

tb\_supplier.setItems(supplierData);

// row action

tb\_supplier.setRowFactory(t -> {

TableRow<Supplier> row = new TableRow<>();

row.setOnMouseClicked(e -> {

// get data from selected row

if (e.getClickCount() == 2 && (!row.isEmpty())) {

Supplier su = tb\_supplier.getSelectionModel().getSelectedItem();

System.out.println("Double click is: " + su.getName());

// set to tf

tf\_id.setText(su.getId());

tf\_name.setText(su.getName());

tf\_supplied\_date.setText(su.getLastSupplied());

}

});

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Supplier su = tb\_supplier.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION, "Are U Sure To Delete " + su.getName() + " ?", ButtonType.YES, ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

//check if this current category is used in product items

//get count

int cateCount = 0;

String getSupplierUsedCount = "SELECT COUNT(\*) FROM productitems, supplier WHERE supplier.id = productitems.supplierid AND supplier.id = '"+su.getId()+"'";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsSupCount = DBInitialize.statement.executeQuery(getSupplierUsedCount);

while(rsSupCount.next()) {

cateCount = rsSupCount.getInt(1);

}

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException

| SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

if(cateCount > 0 ) {

//show can't delete Alert

Alert aal = new Alert(AlertType.ERROR, "Cannot Delete! This Supplier has been used in product items. You can delete the product items that link with this category and try again. Thanks!");

aal.showAndWait();

}else {

//do stuff

String removequery = "DELETE FROM `supplier` WHERE supplier.id = '"+su.getId()+"';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(removequery);

//update table

//update table data

new DBInitialize().DBInitialize();

String queryupdatetable = "SELECT \* FROM `supplier`;";

// get data from db

String query = "SELECT \* FROM `supplier`;";

new DBInitialize().DBInitialize();

new DBInitialize();

supplierData.clear();//clear previous data

ResultSet rs = DBInitialize.statement.executeQuery(queryupdatetable);

while (rs.next()) {

Supplier s = new Supplier();

s.setId(rs.getString(1));

s.setName(rs.getString(2));

s.setLastSupplied(rs.getString(3));

supplierData.add(s);

}

// set data to table

tb\_supplier.refresh();

/\*//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item removed!");

al.showAndWait();\*/

//show alert

Alert al = new Alert(AlertType.INFORMATION, "Item deleted!");

al.showAndWait();

} catch (ClassNotFoundException | SQLException | InstantiationException | IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

});

rowMenu.getItems().addAll( removeItem);

// only display context menu for non-null items:

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty()))

.then(rowMenu)

.otherwise((ContextMenu)null));

return row;

});

}

@FXML

void onNewAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

tf\_id.clear();

tf\_name.clear();

tf\_supplied\_date.clear();

String query = "SELECT supplier.id FROM `supplier` ORDER BY supplier.id DESC LIMIT 1";

String oldid = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while(rs.next()) {

oldid = ""+rs.getInt(1);

}

//count +1 new Id

String newId = ""+(Integer.parseInt(oldid) + 1 );

tf\_id.setText(newId);

//create today date

String pattern = "dd/MM/yyyy";

String todaydate =new SimpleDateFormat(pattern).format(new Date());

System.out.println("today is "+todaydate);

tf\_supplied\_date.setText(todaydate);

}

@FXML

void onAddAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

if(tf\_id.getText().equals("") || tf\_name.getText().equals("") || tf\_supplied\_date.getText().equals("") ||tf\_supplied\_date.getText().length() < 10 ) {

Alert al = new Alert(AlertType.ERROR, "Invalid input or data missing!");

al.showAndWait();

}

else {

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String lastsupplied = tf\_supplied\_date.getText().toString();

try {

String addquery = "INSERT INTO `supplier`(`id`, `companyname`, `lastdatesupplied`) VALUES ("+id+",'"+name+"','"+lastsupplied+"');";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(addquery);

//update table

// get data from db

String query = "SELECT \* FROM `supplier`;";

new DBInitialize().DBInitialize();

new DBInitialize();

supplierData.clear();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

Supplier su = new Supplier();

su.setId(rs.getString(1));

su.setName(rs.getString(2));

su.setLastSupplied(rs.getString(3));

supplierData.add(su);

}

// set data to table

tb\_supplier.refresh();

tf\_id.clear();

tf\_name.clear();

tf\_supplied\_date.clear();

//Alert

Alert al = new Alert(AlertType.INFORMATION, "Item added!");

al.showAndWait();

}//end of try

catch(Exception ex) {

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}//end of else

}

@FXML

void onUpdateAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

if(tf\_id.getText().equals("") || tf\_name.getText().equals("") || tf\_supplied\_date.getText().equals("") ||tf\_supplied\_date.getText().length() < 10 ) {

Alert al = new Alert(AlertType.ERROR, "Invalid input or data missing!");

al.showAndWait();

}

else {

String id = tf\_id.getText().toString();

String name = tf\_name.getText().toString();

String lastsupplied = tf\_supplied\_date.getText().toString();

String upquery = "UPDATE `supplier` SET `companyname`='"+name+"',`lastdatesupplied`='"+lastsupplied+"' WHERE `id`='"+id+"'";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(upquery);

//update table

// get data from db

String query = "SELECT \* FROM `supplier`;";

new DBInitialize().DBInitialize();

new DBInitialize();

supplierData.clear();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while (rs.next()) {

Supplier su = new Supplier();

su.setId(rs.getString(1));

su.setName(rs.getString(2));

su.setLastSupplied(rs.getString(3));

supplierData.add(su);

}

// set data to table

tb\_supplier.refresh();

tf\_id.clear();

tf\_name.clear();

tf\_supplied\_date.clear();

//Alert

Alert al = new Alert(AlertType.INFORMATION, "Item updated!");

al.showAndWait();

}

}//end of else

}

**AdminViewChartController.java**

package controller;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.HashSet;

import java.util.ResourceBundle;

import java.util.Set;

import database.DBInitialize;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.chart.BarChart;

import javafx.scene.chart.CategoryAxis;

import javafx.scene.chart.LineChart;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.PieChart;

import javafx.scene.chart.StackedAreaChart;

import javafx.scene.chart.XYChart;

import javafx.scene.layout.AnchorPane;

public class AdminViewChartController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private AnchorPane ch\_dailySale;

@FXML

private AnchorPane ch\_month;

@FXML

private AnchorPane ch\_category;

@FXML

private AnchorPane ch\_customer\_age;

@FXML

private AnchorPane ch\_cash\_vs\_card;

@FXML

void initialize() throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

assert ch\_dailySale != null : "fx:id=\"ch\_dailySale\" was not injected: check your FXML file 'Admin\_view\_chart.fxml'.";

assert ch\_month != null : "fx:id=\"ch\_month\" was not injected: check your FXML file 'Admin\_view\_chart.fxml'.";

assert ch\_category != null : "fx:id=\"ch\_category\" was not injected: check your FXML file 'Admin\_view\_chart.fxml'.";

assert ch\_customer\_age != null : "fx:id=\"ch\_customer\_age\" was not injected: check your FXML file 'Admin\_view\_chart.fxml'.";

assert ch\_cash\_vs\_card != null : "fx:id=\"ch\_cash\_va\_card\" was not injected: check your FXML file 'Admin\_view\_chart.fxml'.";

showingDailyChart();

showingMonthlyChart();

showingCategoryChart();

showingCashVsCardChart();

showingCustomerAge();

}

private void showingCustomerAge()

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// TODO Auto-generated method stub

// age interval count

int first = 0;

int second = 0;

int third = 0;

int fourth = 0;

int fifth = 0;

// get age data from database

String getAgeQuery = "SELECT `age` FROM `Customer`";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsAge = DBInitialize.statement.executeQuery(getAgeQuery);

while (rsAge.next()) {

int age = Integer.parseInt(rsAge.getString(1));

if (age >= 50) {

// first

first++;

} else if (age >= 41) {

// second

second++;

} else if (age >= 30) {

// third

third++;

} else if (age >= 21) {

// fourth

fourth++;

} else {

// fifth

fifth++;

}

}

// Defining the axes

CategoryAxis xAxis = new CategoryAxis();

// xAxis.setCategories(FXCollections.<String>

// observableArrayList(Arrays.asList("10 - 20", "21 - 29", "30 - 40", "41 - 49",

// "50 and above")));

xAxis.setLabel("Age Interval");

NumberAxis yAxis = new NumberAxis();

yAxis.setLabel("Number of people");

// Creating the Bar chart

BarChart<String, Number> barChart = new BarChart<>(xAxis, yAxis);

barChart.setTitle("Customer Age");

// Prepare XYChart.Series objects by setting data

XYChart.Series<String, Number> series1 = new XYChart.Series<>();

XYChart.Series<String, Number> series2 = new XYChart.Series<>();

XYChart.Series<String, Number> series3 = new XYChart.Series<>();

XYChart.Series<String, Number> series4 = new XYChart.Series<>();

XYChart.Series<String, Number> series5 = new XYChart.Series<>();

series1.getData().add(new XYChart.Data<>("", fifth));

series1.setName("13 - 20");

series2.getData().add(new XYChart.Data<>("", fourth));

series2.setName("21 - 29");

series3.getData().add(new XYChart.Data<>("", third));

series3.setName("30 - 40");

series4.getData().add(new XYChart.Data<>("", second));

series4.setName("41 - 49");

series5.getData().add(new XYChart.Data<>("", first));

series5.setName("50 and above");

System.out.println("50 and above is:...... " + first);

System.out.println("41-50 is:....... " + second);

System.out.println("30-40 is:........ " + third);

System.out.println("21-29:......... " + fourth);

System.out.println("13-20 is:........ " + fifth);

// Setting the data to bar chart

barChart.getData().addAll(series1, series2, series3, series4, series5);

// Setting the data to Line chart

barChart.setAnimated(true);

barChart.animatedProperty();

barChart.setPrefHeight(440);

barChart.setMinHeight(440);

barChart.setMaxHeight(440);

barChart.setPrefWidth(470);

barChart.setMinWidth(470);

barChart.setMaxWidth(470);

barChart.setPrefSize(405, 290);

barChart.setMinSize(405, 290);

barChart.setMaxSize(405, 290);

ch\_customer\_age.getChildren().add(barChart);

System.out.println("Customer age chart is generated ...... ");

}

private void showingCashVsCardChart()

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// TODO Auto-generated method stub

// current date

String pattern = "dd/MM/yyyy";

String today = new SimpleDateFormat(pattern).format(new Date());

System.out.println("Today is ....... " + today);

// current month

String[] tempAry = today.split("/");

String currentMonth = ""+tempAry[1];

System.out.println("current month is ....... " + currentMonth);

// purchase data list

ObservableList<String> purchaseDateList = FXCollections.observableArrayList();

// get data from database

// search by (current) month

String getIdPurchaseQuery = "SELECT DISTINCT `date` FROM `purchase` WHERE purchase.date LIKE '%" + currentMonth + "%'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsPurchase = DBInitialize.statement.executeQuery(getIdPurchaseQuery);

while (rsPurchase.next()) {

String tem = rsPurchase.getString(1);

String[] dateArray = tem.split("/");

purchaseDateList.add(dateArray[0]);

}

//print distinct date

for(int i = 0; i < purchaseDateList.size(); i++) {

System.out.println("Date List :....... "+purchaseDateList.get(i));

}

//purchase count list

ObservableList<Integer> purchaseCountTemp = FXCollections.observableArrayList();

ObservableList<Integer> purchaseCount = FXCollections.observableArrayList();

ObservableList<Integer> transactionCount = FXCollections.observableArrayList();

//get count from db by date

for(int i = 0 ; i < purchaseDateList.size(); i++) {

//purchase count

String countQuery = "SELECT COUNT(\*) FROM `purchase` WHERE purchase.date LIKE '"+purchaseDateList.get(i)+"/"+currentMonth+"%'";

new DBInitialize();

ResultSet rsPCountTemp = DBInitialize.statement.executeQuery(countQuery);

while(rsPCountTemp.next()) {

purchaseCountTemp.add(rsPCountTemp.getInt(1));

}

String countTQuery = "SELECT COUNT(\*) FROM `purchase`, transaction WHERE purchase.date LIKE '"+purchaseDateList.get(i)+"/"+currentMonth+"%' AND purchase.id = transaction.purchaseid";

new DBInitialize();

ResultSet rsTCount = DBInitialize.statement.executeQuery(countTQuery);

while(rsTCount.next()) {

transactionCount.add(rsTCount.getInt(1));

}

purchaseCount.add( purchaseCountTemp.get(i) - transactionCount.get(i));

}//end of for

System.out.println("size of date list is : --------------------- "+purchaseDateList.size());

System.out.println("size of purchase count list is : --------------------- "+purchaseCount.size());

System.out.println("size of transaction count list is : --------------------- "+transactionCount.size());

final NumberAxis xAxis = new NumberAxis(1, 31, 1);

xAxis.setLabel("Days");

final NumberAxis yAxis = new NumberAxis();

yAxis.setLabel("Number of Customer");

final StackedAreaChart<Number, Number> sac = new StackedAreaChart<Number, Number>(xAxis, yAxis);

sac.setTitle("Cash Usage VS Card Usage");

XYChart.Series<Number, Number> seriesCard = new XYChart.Series<Number, Number>();

seriesCard.setName("Card User");

for( int k = 0 ; k < purchaseDateList.size(); k++) {

seriesCard.getData().add(new XYChart.Data<>(Integer.parseInt(purchaseDateList.get(k)), transactionCount.get(k)));

}

/\*seriesCard.getData().add(new XYChart.Data(27, 21));

seriesCard.getData().add(new XYChart.Data(30, 21));

\*/

XYChart.Series<Number, Number> seriesCash = new XYChart.Series<Number, Number>();

seriesCash.setName("Cash User");

for( int t = 0 ; t < purchaseDateList.size(); t++) {

seriesCash.getData().add(new XYChart.Data<>(Integer.parseInt(purchaseDateList.get(t)), purchaseCount.get(t) ));

}

/\*

seriesCash.getData().add(new XYChart.Data(27, 26));

seriesCash.getData().add(new XYChart.Data(31, 26));\*/

sac.getData().addAll(seriesCard, seriesCash);

// Setting the data to Line chart

sac.setAnimated(true);

sac.animatedProperty();

sac.setPrefHeight(555);

sac.setMinHeight(555);

sac.setMaxHeight(555);

sac.setPrefWidth(400);

sac.setMinWidth(400);

sac.setMaxWidth(400);

sac.setPrefSize(555, 280);

sac.setMinSize(555, 280);

sac.setMaxSize(555, 280);

ch\_cash\_vs\_card.getChildren().add(sac);

System.out.println("cash vs card chart is generated ...........");

}

private void showingCategoryChart()

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// TODO Auto-generated method stub

// category array list

ObservableList<String> categoryNameList = FXCollections.observableArrayList();

ObservableList<String> categoryIDList = FXCollections.observableArrayList();

// get category from database

String getCategroyQuery = "SELECT `name`, `id` FROM `productcategory`";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsCategory = DBInitialize.statement.executeQuery(getCategroyQuery);

while (rsCategory.next()) {

categoryNameList.add(rsCategory.getString(1));

categoryIDList.add(rsCategory.getString(2));

}

// category count list

ObservableList<Integer> categoryCount = FXCollections.observableArrayList();

// search count amount by category and plus

for (int i = 0; i < categoryIDList.size(); i++) {

String getCategoryAmountQueyry = "SELECT `count` FROM `productitems` WHERE productitems.categoryid = '"

+ categoryIDList.get(i) + "'";

new DBInitialize();

int cCount = 0;

ResultSet rsCategoryAmount = DBInitialize.statement.executeQuery(getCategoryAmountQueyry);

while (rsCategoryAmount.next()) {

cCount = cCount + rsCategoryAmount.getInt(1);

}

categoryCount.add(cCount);

}

// Preparing ObservbleList object

ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();

// add pie chart data

for (int j = 0; j < categoryNameList.size(); j++) {

pieChartData.add(new PieChart.Data(categoryNameList.get(j), categoryCount.get(j)));

}

/\*

\* new PieChart.Data("Iphone 5S", 13), new PieChart.Data("Samsung Grand", 25),

\* new PieChart.Data("MOTO G", 10), new PieChart.Data("Nokia Lumia", 22), new

\* PieChart.Data("Iphone 6S", 18), new PieChart.Data("Samsung S9", 26), new

\* PieChart.Data("MOTO G", 17), new PieChart.Data("Nokia 8", 23));

\*/

// Creating a Pie chart

PieChart pieChart = new PieChart(pieChartData);

// Setting the title of the Pie chart

pieChart.setTitle("Category Sales");

pieChart.setPrefHeight(400);

pieChart.setMinHeight(400);

pieChart.setMaxHeight(400);

pieChart.setPrefWidth(300);

pieChart.setMinWidth(300);

pieChart.setMaxWidth(300);

pieChart.setPrefSize(420, 300);

pieChart.setMinSize(420, 300);

pieChart.setMaxSize(420, 300);

// setting the direction to arrange the data

pieChart.setClockwise(true);

// Setting the length of the label line

pieChart.setLabelLineLength(50);

// Setting the labels of the pie chart visible

pieChart.setLabelsVisible(true);

// Setting the start angle of the pie chart

pieChart.setStartAngle(180);

pieChart.setAnimated(true);

pieChart.animatedProperty();

ch\_category.getChildren().add(pieChart);

System.out.println("Category chart is generated... ");

}

private void showingMonthlyChart()

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// TODO Auto-generated method stub

// current date

String pattern = "dd/MM/yyyy";

String today = new SimpleDateFormat(pattern).format(new Date());

System.out.println("Today is ....... " + today);

// current year

String[] tempAry = today.split("/");

String currentYear = tempAry[2];

System.out.println("current year is ....... " + currentYear);

// month list

ObservableList<String> monthList = FXCollections.observableArrayList();

ObservableList<String> monthTempList = FXCollections.observableArrayList();

// get data from database

String getMonthQuery = "SELECT DISTINCT `date` FROM `purchase` WHERE purchase.date LIKE '%/" + currentYear

+ "'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsGetMonth = DBInitialize.statement.executeQuery(getMonthQuery);

while (rsGetMonth.next()) {

String dbDate = rsGetMonth.getString(1);

String[] tempA = dbDate.split("/");

String dbMonth = tempA[1];

monthTempList.add(dbMonth);

} // end of while

System.out.println("month temp list 1 ........" + monthTempList.get(0));

// add data from monthtemp to array

String[] monthAry = new String[monthTempList.size()];

for (int m = 0; m < monthTempList.size(); m++) {

monthAry[m] = monthTempList.get(m);

}

// System.out.println("month temp before......."+monthAry);

// delete duplicate data

Set<String> set = new HashSet<String>();

// String[] array = {1,1,2,2,2,3,3,4,5,6,8};

for (String num : monthAry) {

set.add(num);

}

System.out.println("Set .............................." + set);

// convert set to array

String[] setary = set.toArray(new String[set.size()]);

System.out.print("\n set ary is ...................");

for (int b = 0; b < setary.length; b++) {

System.out.print(", " + setary[b]);

}

// add data from setary to real monthlist

for (int a = 0; a < setary.length; a++) {

monthList.add(setary[a]);

}

/\*

\* String current = monthAry[0]; boolean found = false;

\*

\* for (int n = 0; n < monthAry.length ; n++) { if (current == monthAry[n] &&

\* !found) { found = true; } else if (current != monthAry[n]) {

\* System.out.print(" " + current); current = monthAry[n]; found = false; } }

\*/

// System.out.print("month temp after " + monthAry);

// month amount list

ObservableList<String> monthAmountList = FXCollections.observableArrayList();

// search amount by month and plus

for (int j = 0; j < monthList.size(); j++) {

double amount = 0;

String monthAmountQuery = "SELECT `totalamount` FROM `purchase` WHERE purchase.date LIKE '%"

+ monthList.get(j) + "%'";

new DBInitialize();

ResultSet rsMonthAmount = DBInitialize.statement.executeQuery(monthAmountQuery);

while (rsMonthAmount.next()) {

amount = (double) amount + Double.parseDouble(rsMonthAmount.getString(1));

} // end of while

monthAmountList.add("" + amount);

} // end of for

// Defining the x axis

NumberAxis xAxis = new NumberAxis(1, 12, 1);

xAxis.setLabel("Months");

// Defining the y axis

NumberAxis yAxis = new NumberAxis();

yAxis.setLabel("Sale Amount");

// Creating the line chart

LineChart linechart = new LineChart(xAxis, yAxis);

linechart.setTitle("Monthly Sale");

// Prepare XYChart.Series objects by setting data

XYChart.Series series1 = new XYChart.Series();

series1.setName("Monthly sale progress for this year");

for (int k = 0; k < monthList.size(); k++) {

series1.getData().add(

new XYChart.Data(Integer.parseInt(monthList.get(k)), Double.parseDouble(monthAmountList.get(k))));

}

// System.out.println("month "+monthList.get(1)+" amount "+

// monthAmountList.get(1));

/\*

\* series1.getData().add(new XYChart.Data(1, 2000000));

\* series1.getData().add(new XYChart.Data(4, 500000)); series1.getData().add(new

\* XYChart.Data(6, 2000000)); series1.getData().add(new XYChart.Data(9,

\* 1000000)); series1.getData().add(new XYChart.Data(11, 900000));

\* series1.getData().add(new XYChart.Data(12, 4000000));

\*/

// Setting the data to Line chart

linechart.getData().add(series1);

linechart.setAnimated(true);

linechart.animatedProperty();

linechart.setPrefHeight(555);

linechart.setMinHeight(555);

linechart.setMaxHeight(555);

linechart.setPrefWidth(400);

linechart.setMinWidth(400);

linechart.setMaxWidth(400);

linechart.setPrefSize(555, 300);

linechart.setMinSize(555, 300);

linechart.setMaxSize(555, 300);

ch\_month.getChildren().add(linechart);

System.out.println("Monthly Sale chart is generated...");

}

private void showingDailyChart()

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

// TODO Auto-generated method stub

// get data from db and generated to chart data

// current date

String pattern = "dd/MM/yyyy";

String today = new SimpleDateFormat(pattern).format(new Date());

System.out.println("Today is ....... " + today);

// current month

String[] tempAry = today.split("/");

String currentmonth = tempAry[1] + "/" + tempAry[2];

System.out.println("current month is ....... " + currentmonth);

// get data of current month

// get date of this month in purchase

String getDateQuery = "SELECT DISTINCT `date` FROM `purchase` WHERE purchase.date LIKE '%" + currentmonth + "'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsGetData = DBInitialize.statement.executeQuery(getDateQuery);

ObservableList<String> dateAry = FXCollections.observableArrayList();

while (rsGetData.next()) {

dateAry.add(rsGetData.getString(1));

}

// get finaldate array

ObservableList<Integer> finalDateAry = FXCollections.observableArrayList();

for (int i = 0; i < dateAry.size(); i++) {

finalDateAry.addAll(Integer.parseInt(dateAry.get(i).substring(0, 2)));

}

// plus all the amount within that day

ObservableList<Integer> finalAmountAry = FXCollections.observableArrayList();

new DBInitialize().DBInitialize();

for (int j = 0; j < dateAry.size(); j++) {

double totalAmount = 0;

String getAmountQuery = "SELECT `totalamount` FROM `purchase` WHERE purchase.date = '" + dateAry.get(j)

+ "'";

System.out.println("total amount1 is ...... " + totalAmount);

new DBInitialize();

ResultSet rsGetAmountData = DBInitialize.statement.executeQuery(getAmountQuery);

while (rsGetAmountData.next()) {

totalAmount = totalAmount + Double.parseDouble(rsGetAmountData.getString(1));

System.out.println("total amount2 is ...... " + totalAmount);

} // end of while

finalAmountAry.add((int) totalAmount);

System.out.println("total amount3 is ...... " + totalAmount);

} // end of for

// System.out.println("total amount4 is ...... "+totalAmount);

System.out.println("final date ...... " + finalDateAry.get(0));

System.out.println("final amount of date 1 is ...... " + finalAmountAry.get(0));

// Defining the x axis

NumberAxis xAxis = new NumberAxis(1, 31, 2);

xAxis.setLabel("Days");

// Defining the y axis

NumberAxis yAxis = new NumberAxis(50000, 1000000, 100000);

yAxis.setLabel("Sale Amount");

// Creating the line chart

LineChart linechart = new LineChart(xAxis, yAxis);

linechart.setTitle("Daily Sale");

// Prepare XYChart.Series objects by setting data

XYChart.Series series1 = new XYChart.Series();

series1.setName("Daily sale progress within this month");

for (int k = 0; k < dateAry.size(); k++) {

series1.getData().add(new XYChart.Data(finalDateAry.get(k), finalAmountAry.get(k)));

}

/\*

\* series1.getData().add(new XYChart.Data(1970, 15)); series1.getData().add(new

\* XYChart.Data(1980, 30)); series1.getData().add(new XYChart.Data(1990, 60));

\* series1.getData().add(new XYChart.Data(2000, 120)); series1.getData().add(new

\* XYChart.Data(2013, 240)); series1.getData().add(new XYChart.Data(2014, 300));

\*/

// Setting the data to Line chart

linechart.getData().add(series1);

linechart.setAnimated(true);

linechart.animatedProperty();

linechart.setPrefHeight(515);

linechart.setMinHeight(515);

linechart.setMaxHeight(515);

linechart.setPrefWidth(400);

linechart.setMinWidth(400);

linechart.setMaxWidth(400);

linechart.setPrefSize(510, 300);

linechart.setMinSize(510, 300);

linechart.setMaxSize(510, 300);

ch\_dailySale.getChildren().add(linechart);

System.out.println("Daily Sale chart is generated...");

}

}

**AdminViewReportController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import functs.ReportGenerator;

import java.net.URL;

import java.sql.SQLException;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import net.sf.jasperreports.engine.JRException;

public class AdminViewReportController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXButton bt\_daily\_re;

@FXML

private JFXButton bt\_month\_re;

@FXML

private JFXButton bt\_popu\_re;

@FXML

void initialize() {

assert bt\_daily\_re != null : "fx:id=\"bt\_daily\_re\" was not injected: check your FXML file 'Admin\_view\_report.fxml'.";

assert bt\_month\_re != null : "fx:id=\"bt\_month\_re\" was not injected: check your FXML file 'Admin\_view\_report.fxml'.";

assert bt\_popu\_re != null : "fx:id=\"bt\_popu\_re\" was not injected: check your FXML file 'Admin\_view\_report.fxml'.";

}

@FXML

void onDailyAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException, JRException {

new ReportGenerator().generateDailyReport();

}

@FXML

void onMonthlyAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, JRException, SQLException {

new ReportGenerator().generateMonthlyReport();

}

@FXML

void onPopularAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, JRException, SQLException {

new ReportGenerator().generatePopularItem();

}

}

**CardPayController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import common.Common;

import database.DBInitialize;

import functs.ReportGenerator;

import java.io.FileNotFoundException;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.stage.Stage;

import model.CardUser;

import net.sf.jasperreports.engine.JRException;

public class CardPayController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXTextField tf\_purchase\_id;

@FXML

private JFXTextField tf\_total\_amount;

@FXML

private JFXTextField tf\_card\_no;

@FXML

private JFXButton bt\_pay;

@FXML

private JFXTextField tf\_pay\_amount;

String purchasedate;

String cardno;

private double payamount;

double cardamount;

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tf\_purchase\_id != null : "fx:id=\"tf\_purchase\_id\" was not injected: check your FXML file 'Card\_pay.fxml'.";

assert tf\_total\_amount != null : "fx:id=\"tf\_total\_amount\" was not injected: check your FXML file 'Card\_pay.fxml'.";

assert tf\_card\_no != null : "fx:id=\"tf\_card\_no\" was not injected: check your FXML file 'Card\_pay.fxml'.";

assert bt\_pay != null : "fx:id=\"bt\_pay\" was not injected: check your FXML file 'Card\_pay.fxml'.";

assert tf\_pay\_amount != null : "fx:id=\"tf\_pay\_amount\" was not injected: check your FXML file 'Card\_pay.fxml'.";

tf\_card\_no.setEditable(true);

tf\_pay\_amount.setEditable(false);

/\*new Thread(() -> {

try {

ServerSocket ss = new ServerSocket(5000);

System.out.println("Server is running at port : 5000");

while(true) {

Socket s = ss.accept();

DataInputStream inputFromClient = new DataInputStream( s.getInputStream());

String message =inputFromClient.readUTF();

System.out.println("Received from android: " + message);

// inputFromClient.close();

// s.close();

Platform.runLater(()->

tf\_card\_no.setText(""+message));

}

}

catch(Exception ex) {

}

}).start();

\*/

tf\_total\_amount.setText(""+Common.totalAmount);

//get purchase id from previous row

//get previous id and create now id

new DBInitialize().DBInitialize();

String previousgetpurchaseid = " SELECT `id` FROM `purchase` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(previousgetpurchaseid);

String previousid = "";

while(rs.next()) {

previousid = rs.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

tf\_purchase\_id.setText(""+nowid);

Common.purchaseid = nowid;

tf\_card\_no.setOnAction(e->{

cardno = tf\_card\_no.getText().toString();

String getCardInfoQuery = "SELECT \* FROM `Card` WHERE Card.cardnumber = '"+cardno+"';";

CardUser c = new CardUser();

try {

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsc = DBInitialize.statement.executeQuery(getCardInfoQuery);

if(rsc.next()) {

c.setCardno(rsc.getString(1));

c.setCustomrid(rsc.getString(2));

c.setAmount(rsc.getString(3));

c.setLastdateused(rsc.getString(4));

c.setRegisterdate(rsc.getString(5));

c.setExpireddate(rsc.getString(6));

c.setPin(""+rsc.getInt(7));

double totalamount = Double.parseDouble(tf\_total\_amount.getText().toString());

cardamount = Double.parseDouble(c.getAmount());

if(totalamount> cardamount) {

Alert al = new Alert(AlertType.ERROR, "Insufficient Card Balance!");

al.showAndWait();

}

else {

double tominus = (totalamount) \* (0.15);

System.out.println("to minus is : "+tominus);

payamount = totalamount - tominus;

System.out.println("payamount is : "+payamount);

tf\_pay\_amount.setText(""+payamount);

Common.payamount = payamount;

}

///

}else {

Alert al = new Alert(AlertType.ERROR, "Invalid Card!");

al.showAndWait();

}

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException | SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

});

}

@FXML

void onbtPayAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

/\*ObservableList<Sale> saledata = FXCollections.observableArrayList();

saledata = Common.saleitemsdatafromsaletable;\*/

if(tf\_pay\_amount.getText().isEmpty()) {

//do nothing

}else {

String cardinfoforprint = tf\_card\_no.getText().toString();

Common.cardinfo = "A/C XXXXXX XXX XXXX XX "+ cardinfoforprint.substring(cardinfoforprint.length() - 4);

//get purchase id from previous row

//get previous id and create now id

new DBInitialize().DBInitialize();

String previousgetpurchaseid = " SELECT `id` FROM `purchase` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(previousgetpurchaseid);

String previousid = "";

while(rs.next()) {

previousid = rs.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

System.out.println("purchase id for now is : "+nowid);

Common.purchaseid = nowid;

//create today date

String pattern = "dd/MM/yyyy";

purchasedate =new SimpleDateFormat(pattern).format(new Date());

System.out.println("date purchase is "+purchasedate);

//create today current time(purhcase time)

String hour = ""+new Date().getHours();

String min = ""+new Date().getMinutes();

String time = hour+" : "+min;

System.out.println("purhcase time is : "+time);

//db

new DBInitialize().DBInitialize();

new DBInitialize();

String querycreatepurchase = "INSERT INTO `purchase`(`id`, `date`, `time`, `cashierid`, `barcode`, `quantity`, `totalamount` ) "

+ "VALUES ("+Common.purchaseid+", '"+purchasedate+"', '"+time+"',"+Common.cashierrec.getId()+",'"+Common.productids+"','"+Common.productqtys+"','"+payamount+"')";

DBInitialize.statement.executeUpdate(querycreatepurchase);

//get old transaction id and count +1 for new INSERT

String getTranasctionIDQuery = "SELECT `id` FROM `transaction` ORDER BY transaction.id DESC LIMIT 1";

new DBInitialize();

ResultSet rstid = DBInitialize.statement.executeQuery(getTranasctionIDQuery);

String oldtid = "";

while (rstid.next()) {

oldtid = rstid.getString(1);

}

String newtid = ""+(Integer.parseInt(oldtid) + 1);

//add to transaction table

String addtransactionquery = "INSERT INTO `transaction`(`id`, `cashierid`, `cardid`, `purchaseid`, `amount`) VALUES ('"+newtid+"','"+Common.cashierrec.getId()+"','"+cardno+"','"+Common.purchaseid+"','"+payamount+"')";

new DBInitialize();

DBInitialize.statement.executeUpdate(addtransactionquery);

//compute the new card balance amount

double newbalance = cardamount - payamount;

String reductcardmoneyquery = "UPDATE `Card` SET `amount`='"+newbalance+"',`lastuseddate`='"+purchasedate+"' WHERE Card.cardnumber = '"+cardno+"';";

new DBInitialize();

DBInitialize.statement.executeUpdate(reductcardmoneyquery);

//update count and stock amount

//get all the purchase id and count +1 in db

String[] purchasedproductitemsid = Common.productids.split(",");

String[] purchasedproductitemsqty = Common.productqtys.split(",");

for(int i=0; i< purchasedproductitemsid.length; i++) {

int oldcount = 0 ;

int newcount = 0;

//get old count

String getOldCountQuery = "SELECT `count` FROM `productitems` WHERE productitems.barcode = '"+purchasedproductitemsid[i]+"';";

System.out.println("product barcode is "+purchasedproductitemsid[i]);

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsoldc = DBInitialize.statement.executeQuery(getOldCountQuery);

while(rsoldc.next()) {

oldcount = rsoldc.getInt(1);

}

System.out.println("old count is : "+oldcount);

newcount = oldcount + Integer.parseInt(purchasedproductitemsqty[i]);

System.out.println("new count is "+newcount+"purchase qty"+purchasedproductitemsqty[i]);

String updatecountQuery = "UPDATE `productitems` SET `count`= "+newcount+" WHERE productitems.barcode = '"+purchasedproductitemsid[i]+"'";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(updatecountQuery);

//get stock amount

String getstockquery = "SELECT `stockamount` FROM `productitems` WHERE productitems.barcode = '"+purchasedproductitemsid[i]+"';";

String oldstock = "";

new DBInitialize();

ResultSet rsst = DBInitialize.statement.executeQuery(getstockquery);

while(rsst.next()) {

oldstock = rsst.getString(1);

}

String newstock = ""+(Integer.parseInt(oldstock) - Integer.parseInt(purchasedproductitemsqty[i]));

//update stock

String updatestockquery = "UPDATE `productitems` SET `stockamount`='"+newstock+"' WHERE productitems.barcode = '"+purchasedproductitemsid[i]+"'";

new DBInitialize();

DBInitialize.statement.executeUpdate(updatestockquery);

}//end of for

/\*System.out.println("qty is: ");

for(int i = 0; i< purchasedproductitemsqty.length ; i++) {

System.out.println(""+purchasedproductitemsqty[i]);

}

System.out.println("length of id and qty are : "+purchasedproductitemsid.length+" & "+purchasedproductitemsqty.length);\*/

//MainCashierController.clearsaletableitems();

((Stage)bt\_pay.getScene().getWindow()).close();

//alert

Alert trancompleteal = new Alert(AlertType.INFORMATION, "Transaction Complete! "+payamount+" kyats has reduced from the card. Click Ok to print voucher.");

trancompleteal.showAndWait();

//generate report

try {

new ReportGenerator().generatevoucher(Common.saleitemsdatafromsaletable);

//trancompleteal.close();

} catch (JRException | FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}//end of if check tfamount is empty

}

}

**CardRedeemController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXRadioButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Optional;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ToggleGroup;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ButtonType;

import javafx.stage.Stage;

import model.CardUser;

public class CardRedeemController {

@FXML

private JFXTextField tf\_qr\_search;

@FXML

private JFXTextField tf\_customer\_name;

@FXML

private JFXButton bt\_apply;

@FXML

private JFXButton bt\_cancel;

@FXML

private JFXTextField tf\_phone;

@FXML

private JFXTextField tf\_address;

@FXML

private JFXTextField tf\_email;

@FXML

private JFXTextField tf\_pin;

@FXML

private JFXTextField tf\_amount;

@FXML

private JFXTextField tf\_top\_up;

@FXML

private JFXButton bt\_redeem;

@FXML

private JFXTextField tf\_age;

@FXML

private JFXTextField tf\_date\_created;

@FXML

private JFXTextField tf\_expired\_date;

@FXML

private JFXTextField tf\_last\_used;

@FXML

private JFXRadioButton rdo\_male;

@FXML

private ToggleGroup gender;

@FXML

private JFXRadioButton rdo\_female;

private ServerSocket ss;

private Socket s;

private DataInputStream inputFromClient;

private DataOutputStream outputToClient;

String datafromandroid;

private CardUser carduser = new CardUser();

@FXML

void initialize() {

assert tf\_qr\_search != null : "fx:id=\"tf\_qr\_search\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_customer\_name != null : "fx:id=\"tf\_customer\_name\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert bt\_apply != null : "fx:id=\"bt\_apply\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert bt\_cancel != null : "fx:id=\"bt\_cancel\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_phone != null : "fx:id=\"tf\_phone\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_address != null : "fx:id=\"tf\_address\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_email != null : "fx:id=\"tf\_email\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_pin != null : "fx:id=\"tf\_pin\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_amount != null : "fx:id=\"tf\_amount\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_top\_up != null : "fx:id=\"tf\_top\_up\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert bt\_redeem != null : "fx:id=\"bt\_redeem\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_age != null : "fx:id=\"tf\_age\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_date\_created != null : "fx:id=\"tf\_date\_created\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_expired\_date != null : "fx:id=\"tf\_expired\_date\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert tf\_last\_used != null : "fx:id=\"tf\_last\_used\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert rdo\_male != null : "fx:id=\"rdo\_male\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert gender != null : "fx:id=\"gender\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

assert rdo\_female != null : "fx:id=\"rdo\_female\" was not injected: check your FXML file 'Card\_redeem.fxml'.";

rdo\_male.setSelected(true);

if (!tf\_customer\_name.getText().equals("")) {

rdo\_male.setOnAction(e -> {

if (rdo\_male.isSelected()) {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

} else {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

}

carduser.setGender("male");

});

rdo\_female.setOnAction(e -> {

if (rdo\_male.isSelected()) {

rdo\_male.setSelected(false);

rdo\_female.setSelected(true);

} else {

rdo\_male.setSelected(false);

rdo\_female.setSelected(true);

}

carduser.setGender("female");

});

} else {

System.out.println("no action! cause nothing is how edit");

}

/\*

\* Thread thr = new Thread(() -> { try { ss = new ServerSocket(5000);

\* System.out.println("Server is running at port : 5000");

\*

\* while(true) { s = ss.accept(); inputFromClient = new DataInputStream(

\* s.getInputStream()); outputToClient = new

\* DataOutputStream(s.getOutputStream());

\*

\*

\* datafromandroid = inputFromClient.readUTF(); //A8:81:95:8B:1C:AC

\* if(datafromandroid.contains("mac")) {

\* System.out.println("Mac address of andriod is :"+datafromandroid);

\* if(datafromandroid.equals("macA8:81:95:8B:1C:AC")) {

\* System.out.println("mac is working"); outputToClient.writeUTF("Yes"); }else {

\* outputToClient.writeUTF("No"); }

\*

\* } else {

\*

\* System.out.println("Received from android: " + datafromandroid); //

\* inputFromClient.close(); // s.close();

\*

\*

\*

\*

\* Platform.runLater(()-> tf\_qr\_search.setText(""+datafromandroid));

\*

\*

\* //search by qr Platform.runLater(()-> tf\_customer\_name.clear());

\*

\* Platform.runLater(()->tf\_phone.clear());

\*

\* Platform.runLater(()->tf\_email.clear());

\* Platform.runLater(()->tf\_address.clear());

\* Platform.runLater(()->tf\_pin.clear()); Platform.runLater(()->

\* tf\_amount.clear()); Platform.runLater(()->tf\_age.clear());

\* Platform.runLater(()->tf\_date\_created.clear());

\* Platform.runLater(()->tf\_last\_used.clear());

\* Platform.runLater(()->tf\_expired\_date.clear());

\*

\*

\* String cardno = tf\_qr\_search.getText().toString();

\* carduser.setCardno(cardno);

\*

\* new DBInitialize().DBInitialize(); String query =

\* "SELECT `card number`, `type`, `customer id`, `amount`, `lastuseddate`, `registerdate`, `expired date`, `pin`, `description` FROM `card` WHERE `card number`='"

\* +cardno+"';";

\*

\* System.out.println("card no is : "+cardno);

\*

\* new DBInitialize(); ResultSet rs =

\* DBInitialize.statement.executeQuery(query);

\*

\*

\*

\* if(rs.next()) {

\*

\* carduser.setCardno(rs.getString("card number"));

\* carduser.setCustomrid(rs.getString("customer id"));

\* carduser.setAmount(rs.getString("amount"));

\* carduser.setLastdateused(rs.getString("lastuseddate"));

\* carduser.setRegisterdate(rs.getString("registerdate"));

\* carduser.setExpireddate(rs.getString("expired date"));

\* carduser.setPin(rs.getInt("pin")+""); System.out.println("working");

\*

\*

\* //link customer and card (finding customer info) String linkquery =

\* "SELECT `name`, `age`, `gender`, `address`, `phone`, `email` FROM `Customer` WHERE `id` = '"

\* +carduser.getCustomrid()+"';"; new DBInitialize(); ResultSet resultset =

\* DBInitialize.statement.executeQuery(linkquery); if(resultset.next()) {

\* carduser.setName(resultset.getString("name"));

\* carduser.setAge(resultset.getString("age"));

\* carduser.setGender(resultset.getString("gender"));

\* carduser.setAddress(resultset.getString("address"));

\* carduser.setPhone(resultset.getString("phone"));

\* carduser.setEmail(resultset.getString("email"));

\* System.out.println("working");

\*

\*

\*

\* //set data to ui

\*

\* tf\_customer\_name.setText(carduser.getName());

\* tf\_phone.setText(carduser.getPhone()); tf\_email.setText(carduser.getEmail());

\* tf\_address.setText(carduser.getAddress()); tf\_pin.setText(carduser.getPin());

\* tf\_amount.setText(carduser.getAmount()); tf\_age.setText(carduser.getAge());

\* tf\_date\_created.setText(carduser.getRegisterdate());

\* tf\_last\_used.setText(carduser.getLastdateused());

\* tf\_expired\_date.setText(carduser.getExpireddate());

\*

\* if(carduser.getGender().equals("male")) { Platform.runLater(()->

\* rdo\_male.setSelected(true));

\* Platform.runLater(()->rdo\_female.setSelected(false)); } else {

\* Platform.runLater(()->rdo\_female.setSelected(true));

\* Platform.runLater(()->rdo\_male.setSelected(false)); }

\*

\*

\*

\*

\* }//end of second query if else {

\*

\* Platform.runLater(()->tf\_customer\_name.clear());

\* Platform.runLater(()->tf\_phone.clear());

\* Platform.runLater(()->tf\_email.clear());

\* Platform.runLater(()->tf\_address.clear());

\* Platform.runLater(()->tf\_pin.clear());

\* Platform.runLater(()->tf\_amount.clear());

\* Platform.runLater(()->tf\_age.clear());

\* Platform.runLater(()->tf\_date\_created.clear());

\* Platform.runLater(()->tf\_last\_used.clear());

\* Platform.runLater(()->tf\_expired\_date.clear());

\*

\*

\* Alert alert = new Alert(AlertType.ERROR, "No user is found!");

\* Platform.runLater(()->alert.showAndWait());

\*

\* carduser = new CardUser();

\*

\* }

\*

\* }//end of first query if else {

\*

\*

\* Alert alert = new Alert(AlertType.ERROR, "No card is found!");

\* Optional<ButtonType> result = alert.showAndWait(); if (result.get() ==

\* ButtonType.OK) { // delete user

\* Platform.runLater(()->tf\_customer\_name.clear());

\* Platform.runLater(()->tf\_phone.clear());

\* Platform.runLater(()->tf\_email.clear());

\* Platform.runLater(()->tf\_address.clear());

\* Platform.runLater(()->tf\_pin.clear());

\* Platform.runLater(()->tf\_amount.clear());

\* Platform.runLater(()->tf\_age.clear());

\* Platform.runLater(()->tf\_date\_created.clear());

\* Platform.runLater(()->tf\_last\_used.clear());

\* Platform.runLater(()->tf\_expired\_date.clear());

\* System.out.println("okay is working ...."); }

\*

\* carduser = new CardUser();

\*

\* }

\*

\* System.out.println("card ower is : "+carduser.getName() +

\* carduser.getCustomrid()+" last used: "+carduser.getLastdateused());

\*

\*

\* }

\*

\* }} catch(Exception ex) {

\*

\* } }); thr.setPriority(Thread.MAX\_PRIORITY); try {

\* MainCashierController.thcashier.sleep(100000); } catch (InterruptedException

\* e) { // TODO Auto-generated catch block e.printStackTrace(); } thr.start();

\*/

}

@FXML

void onBtCancelAction(ActionEvent event) {

// get a handle to the stage

Stage stage = (Stage) bt\_cancel.getScene().getWindow();

// do what you have to do

stage.close();

}

@FXML

void onBtRedeemAction(ActionEvent event) {

if( tf\_top\_up.getText().equals("") || tf\_top\_up.getText().matches(".\*[a-zA-Z]+.\*")) {

Alert al = new Alert(AlertType.ERROR, "Invalid input");

al.showAndWait();

}

else if (Double.parseDouble(tf\_top\_up.getText().toString()) < 50000 ) {

Alert al = new Alert(AlertType.ERROR, "Amount Is Less than 50000");

al.showAndWait();

} else {

try {

double topupamount = Double.parseDouble(tf\_top\_up.getText().toString());

double existingamount = Double.parseDouble(tf\_amount.getText().toString());

topupamount = topupamount + existingamount;

carduser.setAmount(topupamount + "");

tf\_amount.setText(carduser.getAmount());

System.out.println("Do redeem " + carduser.getAmount());

} catch (Exception ex) {

System.out.println("exception on topup : " + ex.getMessage());

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

} // end of else

tf\_top\_up.clear();

}

@FXML

void onTfQRSearchAction(ActionEvent event)

throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

tf\_customer\_name.clear();

tf\_phone.clear();

tf\_email.clear();

tf\_address.clear();

tf\_pin.clear();

tf\_amount.clear();

tf\_age.clear();

tf\_date\_created.clear();

tf\_last\_used.clear();

tf\_expired\_date.clear();

String cardno = tf\_qr\_search.getText().toString();

carduser.setCardno(cardno);

new DBInitialize().DBInitialize();

String query = "SELECT `cardnumber`, `customerid`, `amount`, `lastuseddate`, `registerdate`, `expireddate`, `pin` FROM `card` WHERE `cardnumber`='"

+ cardno + "';";

System.out.println("card no is : " + cardno);

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

if (rs.next()) {

carduser.setCardno(rs.getString("cardnumber"));

carduser.setCustomrid(rs.getString("customerid"));

carduser.setAmount(rs.getString("amount"));

carduser.setLastdateused(rs.getString("lastuseddate"));

carduser.setRegisterdate(rs.getString("registerdate"));

carduser.setExpireddate(rs.getString("expireddate"));

carduser.setPin(rs.getInt("pin") + "");

System.out.println("working");

// link customer and card (finding customer info)

String linkquery = "SELECT `name`, `age`, `gender`, `address`, `phone`, `email` FROM `Customer` WHERE `id` = '"

+ carduser.getCustomrid() + "';";

new DBInitialize();

ResultSet resultset = DBInitialize.statement.executeQuery(linkquery);

if (resultset.next()) {

carduser.setName(resultset.getString("name"));

carduser.setAge(resultset.getString("age"));

carduser.setGender(resultset.getString("gender"));

carduser.setAddress(resultset.getString("address"));

carduser.setPhone(resultset.getString("phone"));

carduser.setEmail(resultset.getString("email"));

System.out.println("working");

// set data to ui

tf\_customer\_name.setText(carduser.getName());

tf\_phone.setText(carduser.getPhone());

tf\_email.setText(carduser.getEmail());

tf\_address.setText(carduser.getAddress());

tf\_pin.setText(carduser.getPin());

tf\_amount.setText(carduser.getAmount());

tf\_age.setText(carduser.getAge());

tf\_date\_created.setText(carduser.getRegisterdate());

tf\_last\_used.setText(carduser.getLastdateused());

tf\_expired\_date.setText(carduser.getExpireddate());

if (carduser.getGender().equals("male")) {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

} else {

rdo\_female.setSelected(true);

rdo\_male.setSelected(false);

}

} // end of second query if

else {

tf\_customer\_name.clear();

tf\_phone.clear();

tf\_email.clear();

tf\_address.clear();

tf\_pin.clear();

tf\_amount.clear();

tf\_age.clear();

tf\_date\_created.clear();

tf\_last\_used.clear();

tf\_expired\_date.clear();

Alert alert = new Alert(AlertType.ERROR, "No user is found!");

alert.showAndWait();

carduser = new CardUser();

}

} // end of first query if

else {

Alert alert = new Alert(AlertType.ERROR, "No card is found!");

Optional<ButtonType> result = alert.showAndWait();

if (result.get() == ButtonType.OK) {

// delete user

tf\_customer\_name.clear();

tf\_phone.clear();

tf\_email.clear();

tf\_address.clear();

tf\_pin.clear();

tf\_amount.clear();

tf\_age.clear();

tf\_date\_created.clear();

tf\_last\_used.clear();

tf\_expired\_date.clear();

System.out.println("okay is working ....");

}

carduser = new CardUser();

}

System.out.println("card ower is : " + carduser.getName() + carduser.getCustomrid() + " last used: "

+ carduser.getLastdateused());

}

@FXML

void onBtApplyAction(ActionEvent event) throws SQLException {

if (!tf\_customer\_name.getText().equals("") && tf\_phone.getText().contains("09") && tf\_phone.getText().length() > 10 && !tf\_phone.getText().matches(".\*[a-zA-Z]+.\*")

&& !tf\_address.getText().equals("") && tf\_email.getText().contains("mail.com") && tf\_email.getText().contains("@") && !tf\_age.getText().matches(".\*[a-zA-Z]+.\*")

&& Integer.parseInt(tf\_age.getText()) >= 13 ) {

String pattern = "dd/MM/yyyy";

String lastdateused = new SimpleDateFormat(pattern).format(new Date());

System.out.println("last date use is " + lastdateused);

String gender = "";

if (rdo\_male.isSelected()) {

gender = "male";

} else {

gender = "female";

}

carduser.setAmount(tf\_amount.getText());

carduser.setLastdateused(lastdateused);

carduser.setName(tf\_customer\_name.getText());

carduser.setAddress(tf\_address.getText());

carduser.setAge(tf\_age.getText());

carduser.setGender(gender);

carduser.setEmail(tf\_email.getText());

carduser.setPhone(tf\_phone.getText());

// update card data to db

String updatecardquery = " UPDATE `Card` SET `amount`='" + carduser.getAmount() + "',`lastuseddate`='"

+ carduser.getLastdateused() + "' WHERE `cardnumber` = '" + carduser.getCardno() + "' ";

new DBInitialize();

DBInitialize.statement.executeUpdate(updatecardquery);

// update customer data to db

String updatecustomerquery = " UPDATE `Customer` SET `name`='" + carduser.getName() + "',`age`='"

+ carduser.getAge() + "',`gender`='" + gender + "',`address`='" + carduser.getAddress()

+ "',`phone`='" + carduser.getPhone() + "',`email`='" + carduser.getEmail() + "' WHERE `id`= '"

+ carduser.getCustomrid() + "' ";

new DBInitialize();

DBInitialize.statement.executeUpdate(updatecustomerquery);

((Stage) bt\_apply.getScene().getWindow()).close();

;

Alert alert = new Alert(AlertType.INFORMATION, "Information updated.");

alert.showAndWait();

} else {

Alert alert = new Alert(AlertType.ERROR, "Error.");

alert.showAndWait();

}

}

}

**CreateCardController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXDatePicker;

import com.jfoenix.controls.JFXRadioButton;

import com.jfoenix.controls.JFXTextField;

import database.DBInitialize;

import java.io.DataInputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.ToggleGroup;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.Label;

import javafx.stage.Stage;

import model.CardUser;

public class CreateCardController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXTextField tf\_phone;

@FXML

private JFXButton bt\_create;

@FXML

private JFXButton bt\_cancel;

@FXML

private JFXButton bt\_new;

@FXML

private JFXTextField tf\_expired\_date;

@FXML

private JFXTextField tf\_name;

@FXML

private JFXTextField tf\_address;

@FXML

private JFXTextField tf\_email;

@FXML

private JFXTextField tf\_pin;

@FXML

private JFXTextField tf\_amount;

@FXML

private JFXTextField tf\_top\_up;

@FXML

private JFXButton bt\_redeem;

@FXML

private JFXTextField tf\_age;

@FXML

private JFXRadioButton rdo\_male;

@FXML

private ToggleGroup gender;

@FXML

private Label lb\_new\_id;

@FXML

private JFXRadioButton rdo\_female;

@FXML

private JFXTextField tf\_date\_created;

@FXML

private JFXTextField tf\_card\_number;

private CardUser carduser = new CardUser();

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tf\_phone != null : "fx:id=\"tf\_phone\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert bt\_create != null : "fx:id=\"bt\_create\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert bt\_cancel != null : "fx:id=\"bt\_cancel\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_name != null : "fx:id=\"tf\_name\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_address != null : "fx:id=\"tf\_address\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_email != null : "fx:id=\"tf\_email\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_pin != null : "fx:id=\"tf\_pin\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_amount != null : "fx:id=\"tf\_amount\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_top\_up != null : "fx:id=\"tf\_top\_up\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert bt\_redeem != null : "fx:id=\"bt\_redeem\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_age != null : "fx:id=\"tf\_age\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert rdo\_male != null : "fx:id=\"rdo\_male\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert gender != null : "fx:id=\"gender\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert rdo\_female != null : "fx:id=\"rdo\_female\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_expired\_date != null : "fx:id=\"tf\_expired\_date\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_date\_created != null : "fx:id=\"tf\_date\_created\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert tf\_card\_number != null : "fx:id=\"tf\_card\_number\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert lb\_new\_id != null : "fx:id=\"lb\_new\_id\" was not injected: check your FXML file 'Create\_card.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_new\" was not injected: check your FXML file 'Create\_card.fxml'.";

tf\_expired\_date.setEditable(false);

/\* new Thread(() -> {

try {

ServerSocket ss = new ServerSocket(5000);

System.out.println("Server is running at port : 5000");

while(true) {

Socket s = ss.accept();

DataInputStream inputFromClient = new DataInputStream( s.getInputStream());

String message =inputFromClient.readUTF();

System.out.println("Received from android: " + message);

// inputFromClient.close();

// s.close();

Platform.runLater(()->

tf\_card\_number.setText(""+message));

}

}

catch(Exception ex) {

}

}).start();\*/

// tf\_expired\_date.setEditable(false);

//get previous id and create now id

new DBInitialize().DBInitialize();

String previousgetcustomerid = " SELECT `id` FROM `Customer` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(previousgetcustomerid);

String previousid = "";

while(rs.next()) {

previousid = rs.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

carduser.setCustomrid(nowid+"");

lb\_new\_id.setText(nowid+"");

System.out.println("previous id is : "+previousid +" now id is : "+nowid);

carduser.setGender("male");

rdo\_male.setOnAction(e->{

if(rdo\_male.isSelected()) {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

}else {

rdo\_male.setSelected(true);

rdo\_female.setSelected(false);

}

carduser.setGender("male");

});

rdo\_female.setOnAction(e->{

if(rdo\_female.isSelected()) {

rdo\_male.setSelected(false);

rdo\_female.setSelected(true);

}else {

rdo\_male.setSelected(false);

rdo\_female.setSelected(true);

}

carduser.setGender("female");

});

String pattern = "dd/MM/yyyy";

String lastdateused =new SimpleDateFormat(pattern).format(new Date());

System.out.println("last date use is "+lastdateused);

carduser.setLastdateused(lastdateused);

carduser.setRegisterdate(lastdateused);

tf\_date\_created.setText(carduser.getRegisterdate());

String[] expireAry = lastdateused.split("/");

String day = expireAry[0];

String month = expireAry[1];

String year = expireAry[2];

String expireYear = ""+(Integer.parseInt(year) + 3);

tf\_expired\_date.setText(day+"/"+month+"/"+expireYear);

tf\_amount.setText("0");

}

@FXML

void onBtCancelAction(ActionEvent event) {

((Stage)bt\_cancel.getScene().getWindow()).close();

}

@FXML

void onBtCreateAction(ActionEvent event) throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

if(tf\_card\_number.getText().equals("") || tf\_name.getText().equals("") || tf\_phone.getText().equals("") || tf\_address.getText().equals("") || tf\_email.getText().equals("") || tf\_pin.getText().equals("") || tf\_amount.getText().equals("") || tf\_age.getText().equals("")

|| !tf\_phone.getText().contains("09") || tf\_phone.getText().length() < 10 || tf\_phone.getText().matches(".\*[a-zA-Z]+.\*") || !tf\_email.getText().contains("mail.com") || !tf\_email.getText().contains("@") || tf\_age.getText().matches(".\*[a-zA-Z]+.\*")

|| Integer.parseInt(tf\_age.getText()) < 13 ) {

Alert alert = new Alert(AlertType.ERROR, "Error!");

alert.showAndWait();

}

else {

//carduser.setCustomrid(nowid+"");

carduser.setCardno(tf\_card\_number.getText().toString());

carduser.setName(tf\_name.getText().toString());

carduser.setPhone(tf\_phone.getText().toString());

carduser.setAddress(tf\_address.getText().toString());

carduser.setEmail(tf\_email.getText().toString());

carduser.setPin(tf\_pin.getText().toString());

carduser.setAmount(tf\_amount.getText().toString());

carduser.setAge(tf\_age.getText().toString());

//System.out.println("workkkkkkk");

//add data to db

/\* String querycard = " INSERT INTO `Card`(`cardnumber`, `customerid`, `amount`, `lastuseddate`, `registerdate`, `expireddate`, `pin`)"

+ " VALUES ('"+carduser.getCardno()+"', '"+carduser.getCustomrid()+"', '"+carduser.getAmount()+"', '"+carduser.getLastdateused()+"', '"+carduser.getRegisterdate()+"', '"+carduser.getExpireddate()+"', "+carduser.getPin()+") ";

System.out.println("querycard : "+querycard);

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(querycard);

String querycustomer = " INSERT INTO `Customer`(`id`, `name`, `age`, `gender`, `address`, `phone`, `email`) "

+ "VALUES ('"+carduser.getCustomrid()+"', '"+carduser.getName()+"', '"+carduser.getAge()+"', '"+carduser.getGender()+"', '"+carduser.getAddress()+"', '"+carduser.getPhone()+"', '"+carduser.getEmail()+"') ";

System.out.println("querycard : "+querycard);

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(querycustomer);\*/

//create today date

String pattern = "dd/MM/yyyy";

String todaydate =new SimpleDateFormat(pattern).format(new Date());

String addCustomerQuery = "INSERT INTO `Customer`(`id`, `name`, `age`, `gender`, `address`, `phone`, `email`) "

+ "VALUES ('"+carduser.getCustomrid()+"','"+carduser.getName()+"','"+carduser.getAge()+"','"+carduser.getGender()+"','"+carduser.getAddress()+"','"+carduser.getPhone()+"','"+carduser.getEmail()+"')";

String updateCardQuery = "UPDATE `Card` SET `customerid`='"+carduser.getCustomrid()+"',`amount`='"+carduser.getAmount()+"',`lastuseddate`='"+todaydate+"',`registerdate`='"+todaydate+"',`expireddate`='"+carduser.getExpireddate()+"' WHERE `cardnumber`='"+carduser.getCardno()+"'";

new DBInitialize();

DBInitialize.statement.executeUpdate(addCustomerQuery);

DBInitialize.statement.executeUpdate(updateCardQuery);

((Stage)bt\_create.getScene().getWindow()).close();

//show alert

Alert alert = new Alert(AlertType.INFORMATION, "Successful!! Card is created.");

alert.showAndWait();

}

}

@FXML

void onBtRedeemAction(ActionEvent event) {

if( tf\_top\_up.getText().equals("") || tf\_top\_up.getText().matches(".\*[a-zA-Z]+.\*")) {

Alert al = new Alert(AlertType.ERROR, "Invalid input");

al.showAndWait();

}

else if (Double.parseDouble(tf\_top\_up.getText().toString()) < 50000 ) {

Alert al = new Alert(AlertType.ERROR, "Amount Is Less than 50000");

al.showAndWait();

} else {

try{

double topupamount = Double.parseDouble(tf\_top\_up.getText().toString());

double existingamount = Double.parseDouble(tf\_amount.getText().toString());

topupamount = topupamount + existingamount;

carduser.setAmount(topupamount+"");

tf\_amount.setText(carduser.getAmount());

System.out.println("Do redeem "+carduser.getAmount());

}

catch(Exception ex) {

System.out.println("exception on topup : "+ex.getMessage());

Alert al = new Alert(AlertType.ERROR, ""+ex.getMessage());

al.showAndWait();

}

}

tf\_top\_up.clear();

}

@FXML

void onNewAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

new DBInitialize().DBInitialize();

String previousgetcustomerid = " SELECT `id` FROM `Customer` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(previousgetcustomerid);

String previousid = "";

while(rs.next()) {

previousid = rs.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

carduser.setCustomrid(nowid+"");

lb\_new\_id.setText(nowid+"");

System.out.println("previous id is : "+previousid +" now id is : "+nowid);

tf\_phone.clear();

tf\_name.clear();

tf\_address.clear();

tf\_email.clear();

tf\_pin.clear();

tf\_amount.clear();

tf\_top\_up.clear();

tf\_age.clear();

tf\_card\_number.clear();

}

@FXML

void tfCardNumberAction(ActionEvent event) throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

String cardno = tf\_card\_number.getText().toString();

//search card by card no and add info to the tf

String searchCardQuery = "SELECT \* FROM `Card` WHERE Card.cardnumber = '"+cardno+"';";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rssearch = DBInitialize.statement.executeQuery(searchCardQuery);

String datecreated = "";

String expireddate = "";

String lastDateUsed = "";

String customerid = "";

String pin ="";

if(rssearch.next()) {

expireddate = rssearch.getString("expireddate");

//tf\_amount.setText(rssearch.getString("amount"));

lastDateUsed = rssearch.getString("lastuseddate");

datecreated = rssearch.getString("registerdate");

pin = ""+rssearch.getInt("pin");

customerid = rssearch.getString("customerid");

}else {

//show alert

Alert al = new Alert(AlertType.ERROR, " Invalid Card!");

al.showAndWait();

}

if(customerid.equals("")) {

tf\_pin.setText(pin);

}else {

//show alert

Alert al = new Alert(AlertType.ERROR, " Card Already Used!");

al.showAndWait();

}

}

}

**LoginController.java**

package controller;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ResourceBundle;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXPasswordField;

import com.jfoenix.controls.JFXRadioButton;

import com.jfoenix.controls.JFXTextField;

import common.Common;

import database.DBInitialize;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.image.Image;

import javafx.scene.control.ToggleGroup;

import javafx.stage.Stage;

public class LoginController{

@FXML // ResourceBundle that was given to the FXMLLoader

private ResourceBundle resources;

@FXML // URL location of the FXML file that was given to the FXMLLoader

private URL location;

@FXML // fx:id="tf\_id"

private JFXTextField tf\_id; // Value injected by FXMLLoader

@FXML // fx:id="tf\_pass"

private JFXPasswordField tf\_pass; // Value injected by FXMLLoader

@FXML // fx:id="bt\_login"

private JFXButton bt\_login; // Value injected by FXMLLoader

@FXML

private JFXRadioButton bt\_rdo\_admin;

@FXML

private ToggleGroup usertype;

@FXML

private JFXRadioButton bt\_rdo\_cashier;

//db

// private Statement statement;

private ResultSet resultSet;

private String dbQuery;

private String realId ;

private String realPw ;

//strings for data from cashier db

private String name;

private int age;

private String gender;

private String addr;

private String ph;

private String mail;

private String date;

@FXML // This method is called by the FXMLLoader when initialization is complete

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

assert tf\_id != null : "fx:id=\"tf\_id\" was not injected: check your FXML file 'Page\_login.fxml'.";

assert tf\_pass != null : "fx:id=\"tf\_pass\" was not injected: check your FXML file 'Page\_login.fxml'.";

assert bt\_login != null : "fx:id=\"bt\_login\" was not injected: check your FXML file 'Page\_login.fxml'.";

assert bt\_rdo\_admin != null : "fx:id=\"bt\_rdo\_admin\" was not injected: check your FXML file 'Page\_login.fxml'.";

assert usertype != null : "fx:id=\"usertype\" was not injected: check your FXML file 'Page\_login.fxml'.";

assert bt\_rdo\_cashier != null : "fx:id=\"bt\_rdo\_cashier\" was not injected: check your FXML file 'Page\_login.fxml'.";

//database

new DBInitialize().DBInitialize();;

bt\_rdo\_cashier.setSelected(true);

}

/\* private void dbInitialize() throws SQLException, ClassNotFoundException {

// TODO Auto-generated method stub

// Load the JDBC driver

Class.forName("com.mysql.jdbc.Driver"); System.out.println("Driver loaded");

// Connect to a database

Connection connection = DriverManager.getConnection ("jdbc:mysql://localhost:8889/ucsmpos","root","root");

System.out.println("Database connected"); // Create a statement

statement = connection.createStatement();

// Execute a statement

}\*/

@FXML

void bt\_login\_action(ActionEvent event) throws SQLException {

//get dat from 2 text field

try {

String id = tf\_id.getText().toString();

String pw = tf\_pass.getText().toString();

if(bt\_rdo\_admin.isSelected()) {

System.out.println("Admin is seleted");

realId = "abdelheq";

realPw = "admin";

if(id.equals(realId) && pw.equals(realPw)) {

System.out.println("Success!");

//scene transaction

try {

new MainAdmin().start((Stage)bt\_login.getScene().getWindow());

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

else{

//System.out.println("Login Fail! User name or password incorrect! because realid is "+ realId+" and id is "+id+" real password is "+ realPw+ " password is "+pw);

Alert alert = new Alert(AlertType.ERROR, " Login Fail !");

alert.showAndWait();

tf\_id.clear();

tf\_pass.clear();

System.out.println("login fail Error showed");

}

}//end of admin selected

else {

System.out.println("Cashier is seleted");

if(!isInteger(id+"")) {

tf\_id.clear();

tf\_pass.clear();

System.out.println("cashier id enter is String...");

Alert all = new Alert(AlertType.ERROR, "Invalid input!");

all.showAndWait();

}

else {

//if(id.equals(int))

//get user name and password from db

dbQuery = "SELECT \* from cashier where id = "+id+";";

resultSet = DBInitialize.statement.executeQuery(dbQuery);

if(resultSet.next()) {

realId = ""+resultSet.getInt("id");

realPw = resultSet.getString("password");

name = resultSet.getString("name");

age = resultSet.getInt("age");

gender = resultSet.getString("gender");

addr = resultSet.getString("address");

ph = resultSet.getString("phone");

mail = resultSet.getString("email");

date = resultSet.getString("date created");

System.out.println("Id is "+id);

System.out.println("Password is "+pw);

if(id.equals(realId) && pw.equals(realPw)) {

//add cashier info to temp

Common.cashierrec.setId(""+realId);

Common.cashierrec.setPassword(realPw);

Common.cashierrec.setName(name);

Common.cashierrec.setGender(gender);

Common.cashierrec.setAddress(addr);

Common.cashierrec.setAge(""+age);

Common.cashierrec.setPhone(ph);

Common.cashierrec.setEmail(mail);

Common.cashierrec.setDateCreated(date);

System.out.println("Success!");

System.out.println("cashier name is : "+Common.cashierrec.getName());

//scene transaction

try {

new MainCashier().start((Stage)bt\_login.getScene().getWindow());

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

else{

//System.out.println("Login Fail! User name or password incorrect! because realid is "+ realId+" and id is "+id+" real password is "+ realPw+ " password is "+pw);

Alert alert = new Alert(AlertType.ERROR, "Login fail! Incorrect Password");

alert.showAndWait();

tf\_id.clear();

tf\_pass.clear();

System.out.println("login fail Error showed");

}

}//end of rs.next()

else {

Alert alert = new Alert(AlertType.ERROR, "No such user");

alert.showAndWait();

tf\_id.clear();

tf\_pass.clear();

System.out.println("no user Error showed");

//System.out.println("No such user!!");

}

}

}//end of check is Integer

}

catch(NumberFormatException nfe) {

Alert alert = new Alert(AlertType.ERROR, "Invalid input!");

alert.showAndWait();

System.out.println("input Error showed"+nfe.getMessage());

}

}

//for screen transaction from login to cashier panel

public class MainCashier extends Application{

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("/ui/cashier\_main.fxml"));

Scene scene = new Scene(root, 1320, 700);

primaryStage.setScene(scene);

primaryStage.setTitle("Cashier Panel");

//primaryStage.sizeToScene();

primaryStage.setResizable(false);

primaryStage.getIcons().add(new Image("graphic/poslogorect.png"));

primaryStage.setMaximized(false);

primaryStage.show();

}

}

//for screen transaction from login to admin panel

public class MainAdmin extends Application{

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("/ui/Admin\_panel.fxml"));

Scene scene = new Scene(root,1320,700);

primaryStage.setScene(scene);

primaryStage.setTitle("Admin Panel");

//primaryStage.sizeToScene();

primaryStage.setResizable(false);

primaryStage.getIcons().add(new Image("graphic/poslogorect.png"));

primaryStage.setMaximized(false);

primaryStage.show();

}

}

public static boolean isInteger(String s) {

try {

Integer.parseInt(s);

} catch(NumberFormatException e) {

return false;

} catch(NullPointerException e) {

return false;

}

// only got here if we didn't return false

return true;

}

}

**MainAdminController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import java.io.IOException;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.layout.AnchorPane;

import javafx.stage.Stage;

public class MainAdminController {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXButton bt\_cashier;

@FXML

private JFXButton bt\_report;

@FXML

private JFXButton bt\_popular;

@FXML

private JFXButton bt\_chart;

@FXML

private JFXButton bt\_product;

@FXML

private JFXButton bt\_card;

@FXML

private JFXButton bt\_promotion;

@FXML

private JFXButton bt\_category;

@FXML

private JFXButton bt\_customer;

@FXML

private JFXButton bt\_logout;

@FXML

private AnchorPane common\_pane;

@FXML

private JFXButton bt\_supplier;

@FXML

void initialize() {

assert bt\_cashier != null : "fx:id=\"bt\_cashier\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_report != null : "fx:id=\"bt\_report\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_popular != null : "fx:id=\"bt\_popular\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_chart != null : "fx:id=\"bt\_chart\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_product != null : "fx:id=\"bt\_product\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_promotion != null : "fx:id=\"bt\_promotion\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_category != null : "fx:id=\"bt\_category\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_customer != null : "fx:id=\"bt\_customer\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_logout != null : "fx:id=\"bt\_logout\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert common\_pane != null : "fx:id=\"common\_pane\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_supplier != null : "fx:id=\"bt\_supplier\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

assert bt\_card != null : "fx:id=\"bt\_card\" was not injected: check your FXML file 'Admin\_panel.fxml'.";

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/admin\_product.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_product.setDisable(true);

}

// for screen transaction from login to admin panel

public class LoginPg extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("/ui/Page\_login.fxml"));

Scene scene = new Scene(root, 1320, 700);

primaryStage.setScene(scene);

primaryStage.setTitle("Cashier");

//primaryStage.sizeToScene();

primaryStage.setResizable(false);

primaryStage.setMaximized(false);

primaryStage.show();

}

}

@FXML

void onCashierAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_cashier.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_cashier.setDisable(true);

bt\_report.setDisable(false);

bt\_popular.setDisable(false);

bt\_chart.setDisable(false);

bt\_product.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_customer.setDisable(false);

bt\_logout.setDisable(false);

bt\_supplier.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onProductAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_product.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_product.setDisable(true);

bt\_report.setDisable(false);

bt\_popular.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_customer.setDisable(false);

bt\_logout.setDisable(false);

bt\_supplier.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onCategoryAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_category.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_category.setDisable(true);

bt\_report.setDisable(false);

bt\_popular.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_promotion.setDisable(false);

bt\_product.setDisable(false);

bt\_customer.setDisable(false);

bt\_logout.setDisable(false);

bt\_supplier.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onCustomerAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_customer.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_customer.setDisable(true);

bt\_report.setDisable(false);

bt\_popular.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_promotion.setDisable(false);

bt\_product.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_supplier.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onPromotionAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_promotion.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_promotion.setDisable(true);

bt\_report.setDisable(false);

bt\_popular.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_customer.setDisable(false);

bt\_product.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_supplier.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onSupplierAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_supplier.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_supplier.setDisable(true);

bt\_report.setDisable(false);

bt\_popular.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_customer.setDisable(false);

bt\_product.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onPopularAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_popular\_item.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_popular.setDisable(true);

bt\_report.setDisable(false);

bt\_supplier.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_customer.setDisable(false);

bt\_product.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_card.setDisable(false);

}

@FXML

void onManageCardAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_card.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_card.setDisable(true);

bt\_report.setDisable(false);

bt\_supplier.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_customer.setDisable(false);

bt\_product.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_popular.setDisable(false);

}

@FXML

void onReportAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_view\_report.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_report.setDisable(true);

bt\_card.setDisable(false);

bt\_supplier.setDisable(false);

bt\_chart.setDisable(false);

bt\_cashier.setDisable(false);

bt\_customer.setDisable(false);

bt\_product.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_popular.setDisable(false);

}

@FXML

void onChartAction(ActionEvent event) {

AnchorPane pane = null;

try {

pane = FXMLLoader.load(getClass().getResource("/ui/Admin\_view\_chart.fxml"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

common\_pane.getChildren().clear();

common\_pane.getChildren().add(pane);

// bt\_product.setStyle("-fx-background-color : #6e6e6e");

bt\_chart.setDisable(true);

bt\_card.setDisable(false);

bt\_supplier.setDisable(false);

bt\_report.setDisable(false);

bt\_cashier.setDisable(false);

bt\_customer.setDisable(false);

bt\_product.setDisable(false);

bt\_promotion.setDisable(false);

bt\_category.setDisable(false);

bt\_logout.setDisable(false);

bt\_popular.setDisable(false);

}

@FXML

void onLogoutAction(ActionEvent event) {

// scene transaction

try {

new LoginPg().start((Stage) bt\_logout.getScene().getWindow());

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

**MainCashierController.java**

package controller;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXTextField;

import common.Common;

import database.DBInitialize;

import functs.EditingCell;

import functs.ReportGenerator;

import functs.SearchBarcode;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.URL;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

import java.time.temporal.ChronoField;

import java.util.Date;

import java.util.ResourceBundle;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.beans.binding.Bindings;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.geometry.Pos;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ContextMenu;

import javafx.scene.control.Label;

import javafx.scene.control.MenuItem;

import javafx.scene.control.TableCell;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableRow;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.image.Image;

import javafx.scene.control.TableColumn.CellEditEvent;

import javafx.scene.input.KeyEvent;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.util.Callback;

import model.ProductItem;

import model.Sale;

import net.sf.jasperreports.engine.JRException;

public class MainCashierController {

@FXML

private Label lb\_cashier\_name;

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private JFXButton bt\_logout;

@FXML

private JFXTextField tf\_barcode\_search;

@FXML

private JFXButton bt\_new;

@FXML

private JFXButton bt\_create\_card;

@FXML

private TableView<ProductItem> tb\_total\_item;

private TableColumn<ProductItem, String> col\_item\_name;

private TableColumn<ProductItem, String> col\_item\_category;

private TableColumn<ProductItem, String> col\_item\_price;

private TableColumn<ProductItem, String> col\_item\_barcode;

private TableColumn<ProductItem, String> col\_item\_stock;

private ObservableList<ProductItem> data = FXCollections.observableArrayList();

private static ObservableList<Sale> purchasedata = FXCollections.observableArrayList();;

@FXML

private TableView<Sale> tb\_sale;

private TableColumn<Sale, String> col\_purchase\_barcode;

private TableColumn<Sale, String> col\_purchase\_name;

private TableColumn<Sale, String> col\_purchase\_price;

private TableColumn<Sale, Integer> col\_purchase\_quantity;

private TableColumn<Sale, String> col\_purchase\_discount;

private TableColumn<Sale, String> col\_purchase\_totalamount;

@FXML

private JFXButton bt\_pay;

@FXML

private JFXTextField tf\_total;

@FXML

private JFXTextField tf\_pay\_amount;

@FXML

private JFXTextField tf\_change;

@FXML

private JFXTextField tf\_name\_search;

@FXML

private JFXButton btPrint;

@FXML

private Label lb\_slip\_no;

@FXML

private JFXButton bt\_redeem;

private Socket s;

DataInputStream inputFromClient;

DataOutputStream outputToClient;

ServerSocket ss;

private Thread th;

/\*

\* private Thread th1; private Thread th2;

\*/

// public static Thread thcashier;

@FXML

void onLogoutClick(ActionEvent event) {

// scene transaction

try {

new LoginPg().start((Stage) bt\_logout.getScene().getWindow());

th.interrupt();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@FXML

void initialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

// col\_item\_id.setCellValueFactory(new PropertyValueFactory<ProductItem,

// String>("id"));

assert lb\_cashier\_name != null : "fx:id=\"lb\_cashier\_name\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert bt\_logout != null : "fx:id=\"bt\_logout\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tf\_barcode\_search != null : "fx:id=\"tf\_barcode\_search\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tf\_name\_search != null : "fx:id=\"tf\_name\_search\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert bt\_new != null : "fx:id=\"bt\_barcode\_scan\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert bt\_redeem != null : "fx:id=\"bt\_redeem\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert bt\_create\_card != null : "fx:id=\"bt\_create\_card\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tb\_total\_item != null : "fx:id=\"tb\_total\_item\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tb\_sale != null : "fx:id=\"tb\_sale\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert bt\_pay != null : "fx:id=\"bt\_pay\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tf\_total != null : "fx:id=\"tf\_total\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tf\_pay\_amount != null : "fx:id=\"tf\_pay\_amount\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert tf\_change != null : "fx:id=\"tf\_change\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert btPrint != null : "fx:id=\"btPrint\" was not injected: check your FXML file 'cashier\_main.fxml'.";

assert lb\_slip\_no != null : "fx:id=\"lb\_slip\_no\" was not injected: check your FXML file 'cashier\_main.fxml'.";

// set slip number

new DBInitialize().DBInitialize();

String previousgetpurchaseid = " SELECT `id` FROM `purchase` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rsslip = DBInitialize.statement.executeQuery(previousgetpurchaseid);

String previousid = "";

while (rsslip.next()) {

previousid = rsslip.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

lb\_slip\_no.setText("" + nowid);

Common.slipno = "" + nowid;

th = new Thread(() -> {

try {

//tb\_total\_item.refresh();

tf\_name\_search.clear();

tf\_barcode\_search.clear();

ss = new ServerSocket(5000);

System.out.println("Server is running at port : 5000");

while (true) {

s = ss.accept();

inputFromClient = new DataInputStream(s.getInputStream());

outputToClient = new DataOutputStream(s.getOutputStream());

String datafromandriod = inputFromClient.readUTF();

// A8:81:95:8B:1C:AC

System.out.println("Received from android: " + datafromandriod);

// inputFromClient.close();

// s.close();

Platform.runLater(() -> tf\_barcode\_search.setText("" + datafromandriod));

// tb\_total\_item.refresh();

// tb\_total\_item.getItems().clear();

// tb\_total\_item.refresh();

data.clear();

data = SearchBarcode.SearchByBarcode(datafromandriod);

System.out.println("data from function db qr search is : " + data.get(0).getName());

// tb\_total\_item.refresh();

tb\_total\_item.setItems(data);

tb\_total\_item.refresh();

//tb\_total\_item.refresh();

} // end of if

}

catch (Exception ex) {

}

});

th.start();

tf\_total.setAlignment(Pos.BOTTOM\_RIGHT);

tf\_pay\_amount.setAlignment(Pos.BOTTOM\_RIGHT);

tf\_change.setAlignment(Pos.BOTTOM\_RIGHT);

col\_item\_name = new TableColumn<ProductItem, String>("Name");

col\_item\_category = new TableColumn<ProductItem, String>("Category");

col\_item\_price = new TableColumn<ProductItem, String>("Price");

col\_item\_barcode = new TableColumn<ProductItem, String>("Barcode");

col\_item\_stock = new TableColumn<ProductItem, String>("Stock");

col\_item\_name.setMinWidth(200.0);

col\_item\_category.setMinWidth(160.0);

col\_item\_price.setMinWidth(100.0);

col\_item\_barcode.setMinWidth(220.0);

col\_item\_stock.setMinWidth(90.0);

col\_item\_name.setStyle("-fx-font-size: 18");

col\_item\_category.setStyle("-fx-font-size: 18");

col\_item\_price.setStyle("-fx-font-size: 18");

col\_item\_barcode.setStyle("-fx-font-size: 18");

col\_item\_stock.setStyle("-fx-font-size: 18");

col\_item\_name.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("name"));

col\_item\_category.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("categoryname"));

col\_item\_price.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("price"));

col\_item\_barcode.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("barcode"));

col\_item\_stock.setCellValueFactory(new PropertyValueFactory<ProductItem, String>("stockamount"));

tb\_total\_item.getColumns().addAll(col\_item\_barcode, col\_item\_name, col\_item\_category, col\_item\_price,

col\_item\_stock);

// purchase table

col\_purchase\_barcode = new TableColumn<Sale, String>("Barcode");

col\_purchase\_name = new TableColumn<Sale, String>("Name");

col\_purchase\_price = new TableColumn<Sale, String>("Price");

col\_purchase\_quantity = new TableColumn<Sale, Integer>("Quantity");

col\_purchase\_discount = new TableColumn<Sale, String>("Discount");

col\_purchase\_totalamount = new TableColumn<Sale, String>("TotalAmount");

col\_purchase\_barcode.setMinWidth(120.0);

col\_purchase\_name.setMinWidth(90.0);

col\_purchase\_price.setMinWidth(60.0);

col\_purchase\_quantity.setMinWidth(25.0);

col\_purchase\_discount.setMinWidth(25.0);

col\_purchase\_totalamount.setMinWidth(120.0);

col\_purchase\_barcode.setStyle("-fx-font-size: 15");

col\_purchase\_name.setStyle("-fx-font-size: 15");

col\_purchase\_price.setStyle("-fx-font-size: 15");

col\_purchase\_quantity.setStyle("-fx-font-size: 15");

col\_purchase\_discount.setStyle("-fx-font-size: 15");

col\_purchase\_totalamount.setStyle("-fx-font-size: 15");

Callback<TableColumn<Sale, Integer>, TableCell<Sale, Integer>> cellFactory = (

TableColumn<Sale, Integer> param) -> new EditingCell();

col\_purchase\_barcode.setCellValueFactory(new PropertyValueFactory<Sale, String>("barcode"));

col\_purchase\_name.setCellValueFactory(new PropertyValueFactory<Sale, String>("name"));

col\_purchase\_price.setCellValueFactory(new PropertyValueFactory<Sale, String>("unitamount"));

col\_purchase\_quantity.setCellValueFactory(new PropertyValueFactory<Sale, Integer>("quantity"));

col\_purchase\_quantity.setCellFactory(cellFactory);

col\_purchase\_quantity.setOnEditCommit(new EventHandler<CellEditEvent<Sale, Integer>>() {

@Override

public void handle(CellEditEvent<Sale, Integer> t) {

((Sale) t.getTableView().getItems().get(t.getTablePosition().getRow())).setQuantity((t.getNewValue()));

System.out.println("Qty edit Working");

t.getRowValue().setQuantity(t.getNewValue());

double qty = ((Sale) t.getTableView().getItems().get(t.getTablePosition().getRow())).getQuantity();

Double discountpercent = ((Sale) t.getTableView().getItems().get(t.getTablePosition().getRow()))

.getDiscount();

String discountmore = ((Sale) t.getTableView().getItems().get(t.getTablePosition().getRow()))

.getDiscountmore();

String itemmId = ((Sale) t.getTableView().getItems().get(t.getTablePosition().getRow())).getBarcode();

double unitprice = t.getRowValue().getUnitamount();

double total1 = unitprice \* qty;

double total = 0;

// promotion compute

if (discountpercent.equals("0")) {

} else {

double tominuspromotion = total1 \* (discountpercent / 100);

total = total1 - tominuspromotion;

}

if (discountmore.equals("Buy 0 Get 0")) {

} else {

/\*

\* Common.buygetdata.add(((Sale) t.getTableView().getItems().get(

\* t.getTablePosition().getRow()) ).getBarcode());

\*/

}

System.out.println("buy get is " + Common.buygetdata);

t.getRowValue().setTotalamount(total);

tb\_sale.refresh();

tb\_sale.getColumns().get(0).setVisible(false);

tb\_sale.getColumns().get(0).setVisible(true);

int totalall = 0;

for (Sale i : tb\_sale.getItems()) {

totalall += i.getTotalamount();

}

tf\_total.setText("" + totalall);

Common.totalAmount = Double.parseDouble(tf\_total.getText());

System.out.println("Total amount is : " + Common.totalAmount);

// tb\_sale.refresh();

}

});

col\_purchase\_discount.setCellValueFactory(new PropertyValueFactory<Sale, String>("discount"));

/\*

\* col\_purchase\_totalamount.setCellValueFactory(new

\* Callback<CellDataFeatures<Sale, String>, ObservableValue<String>>() {

\*

\*

\*

\* public ObservableValue<String> call(CellDataFeatures<Sale, String> param) {

\*

\*

\* double total = param.getValue().getQuantity() \*

\* Double.parseDouble(param.getValue().getUnitamount());

\*

\*

\* return new SimpleStringProperty(""+total);

\*

\* }

\*

\* });

\*/

col\_purchase\_totalamount.setCellValueFactory(new PropertyValueFactory<Sale, String>("totalamount"));

/\*

\* col\_purchase\_totalamount.setCellValueFactory(cellData -> { Sale data =

\* cellData.getValue(); return Bindings.createDoubleBinding( () -> { try {

\* double price = data.getUnitamount(); double quantity = data.getQuantity();

\* return price \* quantity ; } catch (NumberFormatException nfe) { return 0 ; }

\* }, data.totalamountProperty(), data.quantityProperty() ); });

\*/

tb\_sale.setEditable(true);

tb\_sale.setItems(purchasedata);

tb\_sale.getColumns().addAll(col\_purchase\_barcode, col\_purchase\_name, col\_purchase\_price, col\_purchase\_quantity,

col\_purchase\_discount, col\_purchase\_totalamount);

tb\_sale.refresh();

/\*

\* col\_item\_id.setCellValueFactory(new Callback<CellDataFeatures<ProductItem,

\* String>, ObservableValue<String>>() {

\*

\* public ObservableValue<String> call(CellDataFeatures<ProductItem, String>

\* param) {

\*

\* return new SimpleStringProperty(""); } });

\*/

// set cashier name

lb\_cashier\_name.setText(Common.cashierrec.getName());

// get data from db and set it to table

new DBInitialize().DBInitialize();

String tablequery = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC;";

ResultSet rs = DBInitialize.statement.executeQuery(tablequery);

while (rs.next()) {

ProductItem p = new ProductItem();

p.setBarcode(rs.getString(1));

p.setName(rs.getString(2));

p.setCategoryname(rs.getString(3));

p.setPrice(rs.getString(4));

p.setSuppliername(rs.getString(5));

p.setDateadded(rs.getString(6));

p.setStockamount(rs.getString(7));

p.setExpiredate(rs.getString(8));

data.add(p);

}

tb\_total\_item.setItems(data);

tb\_total\_item.setRowFactory(t -> {

TableRow<ProductItem> row = new TableRow<>();

row.setOnMouseClicked(e -> {

// get data from selected row

// ProductItem productItem =

// tb\_total\_item.getSelectionModel().getSelectedItem();

// System.out.println("Select row is : "+productItem.getName());

if (e.getClickCount() == 2 && (!row.isEmpty())) {

String dispercentage = "0";

String dismore = "Buy 0 Get 0";

ProductItem product = tb\_total\_item.getSelectionModel().getSelectedItem();

System.out.println("Double click is: " + product.getName());

// get discount form db

String discountQuery = "SELECT promotion.percentage, promotion.description FROM `promotion` WHERE promotion.productid = '"

+ product.getBarcode() + "';";

try {

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsd = DBInitialize.statement.executeQuery(discountQuery);

if (rsd.next()) {

dispercentage = rsd.getString(1);

dismore = rsd.getString(2);

} else {

System.out.println("no discount");

}

System.out

.println("percentage from db is ::::" + dispercentage + " &&&& more is ::::" + dismore);

product.setDiscount(dispercentage);

product.setDiscountmore(dismore);

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException

| SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

// create virtual sale item

Sale sa = new Sale();

double total = 0;

// promotion compute

if (dispercentage.equals("0")) {

sa.setTotalamount(Double.parseDouble(product.getPrice()));

System.out.println("dispercent 0 is working");

} else {

System.out.println("unit price is :::" + product.getPrice());

System.out.println("discount percnet is :::" + dispercentage);

double tominuspromotion = Double.parseDouble(product.getPrice())

\* (Double.parseDouble(dispercentage) / 100);

total = Double.parseDouble(product.getPrice()) - tominuspromotion;

System.out.println("unit amount after discount is : " + total);

sa.setTotalamount(Double.parseDouble(total + ""));

System.out.println("dispercent compute is working");

}

if (dismore.equals("Buy 0 Get 0")) {

// do nothing

} else {

Common.buygetdata.add(product.getBarcode());

}

System.out.println("buy get is :::::::::::::" + Common.buygetdata);

// set sale data

sa.setBarcode("" + product.getBarcode());

sa.setName(product.getName());

sa.setQuantity(1);

sa.setUnitamount(Double.parseDouble(product.getPrice()));

sa.setDiscount(Double.parseDouble(product.getDiscount()));

sa.setDiscountmore(product.getDiscountmore());

// double totalAmount = count \* Double.parseDouble(sale.getUnitamount());

// sa.setTotalamount(Double.parseDouble(""+product.getPrice()));

purchasedata.add(sa);

tb\_sale.refresh();

int totalall = 0;

for (Sale i : tb\_sale.getItems()) {

totalall += i.getTotalamount();

}

tf\_total.setText("" + totalall);

Common.totalAmount = Double.parseDouble(tf\_total.getText());

System.out.println("Total amount is : " + Common.totalAmount);

}

});

return row;

});

tb\_sale.setRowFactory(t -> {

TableRow<Sale> r = new TableRow<>();

r.setOnMouseClicked(e -> {

// get data from selected row

// ProductItem productItem =

// tb\_total\_item.getSelectionModel().getSelectedItem();

// System.out.println("Select row is : "+productItem.getName());

if (e.getClickCount() == 2 && (!r.isEmpty())) {

Sale sale = tb\_sale.getSelectionModel().getSelectedItem();

System.out.println("sale Double click is: " + sale.getName());

}

tb\_sale.refresh();

});

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Delete");

removeItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Sale s = tb\_sale.getSelectionModel().getSelectedItem();

Alert alert = new Alert(AlertType.CONFIRMATION, "Are U Sure To Delete " + s.getName() + " ?",

ButtonType.YES, ButtonType.NO);

alert.showAndWait();

if (alert.getResult() == ButtonType.YES) {

// do stuff

// reduce all total ammount

Sale se = purchasedata.get(tb\_sale.getSelectionModel().getFocusedIndex());

Common.totalAmount = Common.totalAmount - se.getTotalamount();

tf\_total.setText("" + Common.totalAmount);

tf\_pay\_amount.clear();

tf\_change.clear();

purchasedata.remove(tb\_sale.getSelectionModel().getFocusedIndex());

tb\_sale.refresh();

}

}

});

rowMenu.getItems().addAll(removeItem);

// only display context menu for non-null items:

r.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(r.itemProperty())).then(rowMenu).otherwise((ContextMenu) null));

return r;

});

// display total

// tf\_total.textProperty().setValue();

// tf\_total.setText(""+total);

}

@FXML

void onbtRedeemClick(ActionEvent event) {

Stage sg = new Stage();

FXMLLoader root = new FXMLLoader(getClass().getResource("/ui/Card\_redeem.fxml"));

Scene scene = null;

try {

scene = new Scene(((Parent) root.load()), 600, 450);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

sg.setScene(scene);

sg.setResizable(false);

sg.setMaximized(false);

sg.setTitle("Redeem Cash Card");

sg.getIcons().add(new Image("graphic/poslogorect.png"));

sg.initModality(Modality.APPLICATION\_MODAL);

sg.show();

}

// for screen transaction from login to admin panel

public class LoginPg extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("/ui/Page\_login.fxml"));

Scene scene = new Scene(root, 1320, 700);

primaryStage.setScene(scene);

primaryStage.setTitle("Cashier");

// primaryStage.sizeToScene();

primaryStage.setResizable(false);

primaryStage.setMaximized(false);

primaryStage.show();

}

}

// search product item table by barcode or name

@FXML

void tfTypeSearchAction(KeyEvent event) throws ClassNotFoundException, SQLException, InterruptedException {

th.sleep(1);

tf\_name\_search.clear();

/\*

\* if(!th.isInterrupted()) { th.start();

\* System.out.println("------------------------------- xD"); }else {

\* th.interrupt(); System.out.println("------------------------------- xP"); }

\*/

String searchKey = tf\_barcode\_search.getText().toString();

System.out.println("key entered is : " + searchKey);

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id AND productitems.barcode LIKE '"

+ searchKey + "%'";

// new DBInitialize().DBInitialize();

System.out.println("working");

try {

// ResultSet rs = st.executeQuery("SELECT \* FROM USER");

ResultSet rs = DBInitialize.statement.executeQuery(query);

ObservableList<ProductItem> row = FXCollections.observableArrayList();

while (rs.next()) {

ProductItem p = new ProductItem();

p.setBarcode(rs.getString(1));

p.setName(rs.getString(2));

p.setCategoryname(rs.getString(3));

p.setPrice(rs.getString(4));

p.setSuppliername(rs.getString(5));

p.setDateadded(rs.getString(6));

p.setStockamount(rs.getString(7));

p.setExpiredate(rs.getString(8));

row.add(p);

}

tb\_total\_item.setItems(row);

// System.out.println("working1"+data);

// tb\_total\_item.getItems().clear();

// tb\_total\_item.setItems(data);

System.out.println("working2");

// data.getItems().addAll(row);

} catch (SQLException ex) {

}

}

@FXML

void tfNameSearchAction(KeyEvent event) throws InterruptedException {

th.sleep(1);

tf\_barcode\_search.clear();

/\*

\* if(!th.isInterrupted()) { th.start();

\* System.out.println("------------------------------- xD"); }else {

\* th.interrupt(); System.out.println("------------------------------- xP"); }

\*/

String searchKey = tf\_name\_search.getText().toString();

System.out.println("key entered is : " + searchKey);

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id AND productitems.name LIKE '"

+ searchKey + "%'";

// new DBInitialize().DBInitialize();

System.out.println("working");

try {

// ResultSet rs = st.executeQuery("SELECT \* FROM USER");

ResultSet rs = DBInitialize.statement.executeQuery(query);

ObservableList<ProductItem> row = FXCollections.observableArrayList();

while (rs.next()) {

ProductItem p = new ProductItem();

p.setBarcode(rs.getString(1));

p.setName(rs.getString(2));

p.setCategoryname(rs.getString(3));

p.setPrice(rs.getString(4));

p.setSuppliername(rs.getString(5));

p.setDateadded(rs.getString(6));

p.setStockamount(rs.getString(7));

p.setExpiredate(rs.getString(8));

row.add(p);

}

tb\_total\_item.setItems(row);

// System.out.println("working1"+data);

// tb\_total\_item.getItems().clear();

// tb\_total\_item.setItems(data);

System.out.println("working2");

// data.getItems().addAll(row);

} catch (SQLException ex) {

}

}

@FXML

void onEnterButtonClick(ActionEvent event) {

if (tf\_pay\_amount.getText().equals("")) {

Alert al = new Alert(AlertType.ERROR, "No Input!");

al.showAndWait();

} else if (tf\_pay\_amount.getText().matches(".\*[a-zA-Z]+.\*")) {

Alert al = new Alert(AlertType.ERROR, "Please input the right amount in number!");

al.showAndWait();

} else if (Double.parseDouble(tf\_pay\_amount.getText()) < Common.totalAmount) {

// do nothing

Alert al = new Alert(AlertType.ERROR, "Invalid amount!");

al.showAndWait();

tf\_pay\_amount.clear();

} else {

try {

Common.payamount = Double.parseDouble(tf\_pay\_amount.getText());

Common.change = Common.payamount - Common.totalAmount;

tf\_change.setText("" + Common.change);

System.out.println("Total Amount is: " + Common.totalAmount);

System.out.println("Pay Amount is: " + Common.payamount);

System.out.println("Change is: " + Common.change);

} catch (Exception ex) {

System.out.println("Error in payamount: " + ex.getMessage());

}

} // end of else

}

// customer number textfield

public class NumberTextField extends TextField {

@Override

public void replaceText(int start, int end, String text) {

if (validate(text)) {

super.replaceText(start, end, text);

}

}

@Override

public void replaceSelection(String text) {

if (validate(text)) {

super.replaceSelection(text);

}

}

private boolean validate(String text) {

return text.matches("[0-9]\*");

}

}

@FXML

void onbtCreateCardClick(ActionEvent event) {

Stage sg = new Stage();

FXMLLoader root = new FXMLLoader(getClass().getResource("/ui/Create\_card.fxml"));

Scene scene = null;

try {

scene = new Scene(((Parent) root.load()), 600, 450);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

sg.setScene(scene);

sg.setResizable(true);

sg.setMaximized(false);

sg.setTitle("Create Card");

sg.getIcons().add(new Image("graphic/poslogorect.png"));

sg.initModality(Modality.APPLICATION\_MODAL);

sg.show();

}

@FXML

void onBtPayClick(ActionEvent event) {

// get all data from table and add to oberable list

if (tf\_total.getText().equals("")) {

System.out.println("Do nothing");

Alert al = new Alert(AlertType.ERROR, "No item to sale");

al.showAndWait();

} else {

ObservableList<Sale> saledata = FXCollections.observableArrayList();

Common.saleitemsdatafromsaletable = saledata;// for generating report in pay action

saledata = tb\_sale.getItems();

StringBuilder productid = new StringBuilder();

StringBuilder qty = new StringBuilder();

for (int i = 0; i < saledata.size(); i++) {

productid.append(saledata.get(i).getBarcode());

productid.append(",");

qty.append(saledata.get(i).getQuantity());

qty.append(",");

}

Common.totalAmount = Double.parseDouble(tf\_total.getText());

// Common.saleitemsdatafromsaletable = saledata;

Common.productids = productid.toString();

Common.productqtys = qty.toString();

Common.saleitemsdatafromsaletable = saledata;// for generating report in pay action

Common.paidtype = "Card";

System.out.println("appended qty is : " + qty.toString());

// open new stage

Stage sg = new Stage();

FXMLLoader root = new FXMLLoader(getClass().getResource("/ui/Card\_pay.fxml"));

Scene scene = null;

try {

scene = new Scene(((Parent) root.load()), 550, 330);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

sg.setScene(scene);

sg.setResizable(true);

sg.setMaximized(false);

sg.setTitle("Card Payment");

sg.getIcons().add(new Image("graphic/poslogorect.png"));

sg.initModality(Modality.APPLICATION\_MODAL);

sg.show();

}

}

@FXML

void onbtPrintClick(ActionEvent event)

throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

if (tf\_change.getText().equals("") || tf\_pay\_amount.getText().equals("")) {

// do nothing

tf\_change.clear();

tf\_pay\_amount.clear();

/\*

\* Common.change = 0; Common.payamount = 0; Common.totalAmount =

\* Double.parseDouble(tf\_total.getText().toString());

\*/

Alert al = new Alert(AlertType.ERROR, "Please enter pay amount!");

al.showAndWait();

} else {

ObservableList<Sale> saledata = FXCollections.observableArrayList();

saledata = tb\_sale.getItems();

StringBuilder productid = new StringBuilder();

StringBuilder qty = new StringBuilder();

for (int i = 0; i < saledata.size(); i++) {

productid.append(saledata.get(i).getBarcode());

productid.append(",");

qty.append(saledata.get(i).getQuantity());

qty.append(",");

}

// add purchase items id to purchase table

System.out.println("Product barcodes are : " + productid);

System.out.println("Product quantities are : " + qty);

// get purchase id from previous row

// get previous id and create now id

try {

new DBInitialize().DBInitialize();

String previousgetpurchaseid = " SELECT `id` FROM `purchase` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(previousgetpurchaseid);

String previousid = "";

while (rs.next()) {

previousid = rs.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

System.out.println("purchase id for now is : " + nowid);

Common.purchaseid = nowid;

// create today date

String pattern = "dd/MM/yyyy";

String purchasedate = new SimpleDateFormat(pattern).format(new Date());

System.out.println("date purchase is " + purchasedate);

// create today current time(purhcase time)

String myDateString = "13:24:40";

LocalTime localTime = LocalTime.parse(myDateString, DateTimeFormatter.ofPattern("HH:mm:ss"));

int hour = localTime.get(ChronoField.CLOCK\_HOUR\_OF\_DAY);

int minute = localTime.get(ChronoField.MINUTE\_OF\_HOUR);

int second = localTime.get(ChronoField.SECOND\_OF\_MINUTE);

String currenttime = hour + ":" + minute + ":" + second + "";

System.out.println("purhcase time is : " + currenttime);

// Common.currenttime = currenttime;

Common.totalAmount = Double.parseDouble(tf\_total.getText());

// db

new DBInitialize().DBInitialize();

new DBInitialize();

String querycreatepurchase = "INSERT INTO `purchase`(`id`, `date`, `time`, `cashierid`, `barcode`, `quantity`, `totalamount` ) "

+ "VALUES (" + Common.purchaseid + ", '" + purchasedate + "', '" + currenttime + "','"

+ Common.cashierrec.getId() + "','" + productid + "','" + qty + "','" + Common.totalAmount

+ "')";

DBInitialize.statement.executeUpdate(querycreatepurchase);

// update count and stock amount

// get all the purchase id and count +1 in db

String[] purchasedproductitemsid = productid.toString().split(",");

String[] purchasedproductitemsqty = qty.toString().split(",");

for (int i = 0; i < purchasedproductitemsid.length; i++) {

int oldcount = 0;

int newcount = 0;

// get old count

String getOldCountQuery = "SELECT `count` FROM `productitems` WHERE productitems.barcode = '"

+ purchasedproductitemsid[i] + "';";

System.out.println("product barcode is " + purchasedproductitemsid[i]);

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsoldc = DBInitialize.statement.executeQuery(getOldCountQuery);

while (rsoldc.next()) {

oldcount = rsoldc.getInt(1);

}

System.out.println("old count is : " + oldcount);

newcount = oldcount + Integer.parseInt(purchasedproductitemsqty[i]);

System.out.println("new count is " + newcount + "purchase qty" + purchasedproductitemsqty[i]);

String updatecountQuery = "UPDATE `productitems` SET `count`= " + newcount

+ " WHERE productitems.barcode = '" + purchasedproductitemsid[i] + "'";

new DBInitialize().DBInitialize();

new DBInitialize();

DBInitialize.statement.executeUpdate(updatecountQuery);

// get stock amount

String getstockquery = "SELECT `stockamount` FROM `productitems` WHERE productitems.barcode = '"

+ purchasedproductitemsid[i] + "';";

String oldstock = "";

new DBInitialize();

ResultSet rsst = DBInitialize.statement.executeQuery(getstockquery);

while (rsst.next()) {

oldstock = rsst.getString(1);

}

String newstock = "" + (Integer.parseInt(oldstock) - Integer.parseInt(purchasedproductitemsqty[i]));

// update stock

String updatestockquery = "UPDATE `productitems` SET `stockamount`='" + newstock

+ "' WHERE productitems.barcode = '" + purchasedproductitemsid[i] + "'";

new DBInitialize();

DBInitialize.statement.executeUpdate(updatestockquery);

} // end of for

/\*

\* System.out.println("qty is: "); for(int i = 0; i<

\* purchasedproductitemsqty.length ; i++) {

\* System.out.println(""+purchasedproductitemsqty[i]); }

\* System.out.println("length of id and qty are : "+purchasedproductitemsid.

\* length+" & "+purchasedproductitemsqty.length);

\*/

// MainCashierController.clearsaletableitems();

// ((Stage)bt\_pay.getScene().getWindow()).close();

// alert

Common.payamount = Double.parseDouble(tf\_pay\_amount.getText().toString());

Common.change = Double.parseDouble(tf\_change.getText().toString());

Common.paidtype = "Cash";

Common.cardinfo = "";

tf\_total.clear();

tf\_pay\_amount.clear();

tf\_change.clear();

// update in stock table

// get data from db and set it to table

new DBInitialize().DBInitialize();

data.clear();

String tablequery = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC;";

ResultSet rsu = DBInitialize.statement.executeQuery(tablequery);

while (rsu.next()) {

ProductItem pr = new ProductItem();

pr.setBarcode(rsu.getString(1));

pr.setName(rsu.getString(2));

pr.setCategoryname(rsu.getString(3));

pr.setPrice(rsu.getString(4));

pr.setSuppliername(rsu.getString(5));

pr.setDateadded(rsu.getString(6));

pr.setStockamount(rsu.getString(7));

pr.setExpiredate(rsu.getString(8));

data.add(pr);

}

tb\_total\_item.refresh();

/\*

\* //alert Alert trancompleteal = new Alert(AlertType.INFORMATION,

\* "Please wait! Voucher is being generated..."); trancompleteal.showAndWait();

\*/

// Common.payamount =

// generate report

try {

new ReportGenerator().generatevoucher(saledata);

// trancompleteal.close();

} catch (JRException | FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

// clear sale data

purchasedata.clear();

tb\_sale.refresh();

Common.totalAmount = 0;

tf\_total.clear();

// set slip number

new DBInitialize().DBInitialize();

String getpreviouspurchaseid = " SELECT `id` FROM `purchase` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rsslipno = DBInitialize.statement.executeQuery(previousgetpurchaseid);

String olddid = "";

while (rsslipno.next()) {

olddid = rsslipno.getString("id");

}

int nowwid = Integer.parseInt(olddid) + 1;

lb\_slip\_no.setText("" + nowwid);

Common.slipno = "" + nowwid;

Common.buygetdata.clear();

Common.buygetitem = "";

Common.buygetpromo = "";

Common.totalAmount = 0;

Common.change = 0;

Common.payamount = 0;

} // end of try

catch (Exception ex) {

Alert al = new Alert(AlertType.ERROR, "" + ex.getMessage());

al.showAndWait();

}

}

}

@FXML

void onbtNewClick(ActionEvent event)

throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

Common.buygetdata.clear();

Common.saleitemsdatafromsaletable.clear();

// set slip number

new DBInitialize().DBInitialize();

String previousgetpurchaseid = " SELECT `id` FROM `purchase` ORDER BY `id` DESC LIMIT 1 ";

new DBInitialize();

ResultSet rsslip = DBInitialize.statement.executeQuery(previousgetpurchaseid);

String previousid = "";

while (rsslip.next()) {

previousid = rsslip.getString("id");

}

int nowid = Integer.parseInt(previousid) + 1;

lb\_slip\_no.setText("" + nowid);

Common.slipno = "" + nowid;

// clear sale data

purchasedata.clear();

tb\_sale.refresh();

Common.totalAmount = 0;

tf\_total.clear();

tf\_pay\_amount.clear();

tf\_change.clear();

// update instock table

// get data from db and set it to table

new DBInitialize().DBInitialize();

data.clear();

String tablequery = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.barcode DESC;";

ResultSet rs = DBInitialize.statement.executeQuery(tablequery);

while (rs.next()) {

ProductItem p = new ProductItem();

p.setBarcode(rs.getString(1));

p.setName(rs.getString(2));

p.setCategoryname(rs.getString(3));

p.setPrice(rs.getString(4));

p.setSuppliername(rs.getString(5));

p.setDateadded(rs.getString(6));

p.setStockamount(rs.getString(7));

p.setExpiredate(rs.getString(8));

data.addAll(p);

}

tb\_total\_item.refresh();

//tf\_name\_search.clear();

//tf\_barcode\_search.clear();

}

}

**database**

**ConnectionUtils.java**

package database;

import java.sql.Connection;

import java.sql.SQLException;

public class ConnectionUtils {

public static Connection getConnection() throws SQLException,

ClassNotFoundException {

// Using Oracle

// You may be replaced by other Database.

return MySQLconnUtils.getMySQLConnection();

}

//

// Test Connection ...

//

/\* public static void main(String[] args) throws SQLException,

ClassNotFoundException {

System.out.println("Get connection ... ");

// Get a Connection object

Connection conn = ConnectionUtils.getConnection();

System.out.println("Get connection " + conn);

System.out.println("Done!");

}\*/

}

**DBInitialize.java**

package database;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class DBInitialize {

public static Statement statement;

public void DBInitialize() throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException {

// Load the JDBC driver

Class.forName("com.mysql.jdbc.Driver").newInstance();

System.out.println("Driver loaded");

// Connect to a database

Connection connection = DriverManager.getConnection ("jdbc:mysql://localhost:3306/mxm","root","");

System.out.println("Database connected"); // Create a statement

statement = connection.createStatement();

// Execute a statement

}

}

**MySQLconnUtils.java**

package database;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class MySQLconnUtils {

public static Connection getMySQLConnection()

throws ClassNotFoundException, SQLException {

String hostName = "localhost";

String dbName = "mxm";

String userName = "root";

String password = "root";

return getMySQLConnection(hostName, dbName, userName, password);

}

public static Connection getMySQLConnection(String hostName, String dbName,

String userName, String password) throws SQLException,

ClassNotFoundException {

Class.forName("com.mysql.jdbc.Driver");

System.out.println("Get qq ... ");

String connectionURL = "jdbc:mysql://" + hostName + ":3306/" + dbName;

Connection conn = DriverManager.getConnection(connectionURL, userName,

password);

return conn;

}

}

**functs**

**EditingCell.java**

package functs;

import javafx.beans.value.ObservableValue;

import javafx.scene.control.TableCell;

import javafx.scene.control.TextField;

import model.Sale;

public class EditingCell extends TableCell<Sale, Integer> {

private TextField textField;

public EditingCell() {

}

@Override

public void startEdit() {

if (!isEmpty()) {

super.startEdit();

createTextField();

setText(null);

setGraphic(textField);

textField.selectAll();

}

}

@Override

public void cancelEdit() {

super.cancelEdit();

setText("" + getItem());

setGraphic(null);

}

@Override

public void updateItem(Integer item, boolean empty) {

super.updateItem(item, empty);

if (!empty) {

int row = getIndex();

Sale sale = getTableView().getItems().get(row);

if (empty) {

setText("" + item);

setGraphic(null);

} else {

if (isEditing()) {

if (textField != null) {

textField.setText(getString());

// setGraphic(null);

}

setText(null);

setGraphic(textField);

} else {

setText(getString());

setGraphic(null);

}

}

}

}

private void createTextField() {

textField = new TextField(getString());

textField.setMinWidth(this.getWidth() - this.getGraphicTextGap() \* 2);

textField.setOnAction((e) -> commitEdit(Integer.parseInt(textField.getText())));

textField.focusedProperty()

.addListener((ObservableValue<? extends Boolean> observable, Boolean oldValue, Boolean newValue) -> {

if (!newValue) {

System.out.println("Commiting " + textField.getText());

commitEdit(Integer.parseInt(textField.getText()));

}

});

}

private String getString() {

return getItem() == null ? "" : "" + getItem();

}

}

**ReportGenerator.java**

package functs;

import java.io.File;

import java.io.FileNotFoundException;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import org.apache.log4j.BasicConfigurator;

import common.Common;

import database.DBInitialize;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import model.DailyReport;

import model.DailyTable;

import model.DayItemTable;

import model.ProductItem;

import model.Sale;

import net.sf.jasperreports.engine.JREmptyDataSource;

import net.sf.jasperreports.engine.JRException;

import net.sf.jasperreports.engine.JasperFillManager;

import net.sf.jasperreports.engine.JasperPrint;

import net.sf.jasperreports.engine.JasperReport;

import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

import net.sf.jasperreports.engine.export.JRPdfExporter;

import net.sf.jasperreports.engine.util.JRLoader;

import net.sf.jasperreports.export.ExporterInput;

import net.sf.jasperreports.export.OutputStreamExporterOutput;

import net.sf.jasperreports.export.SimpleExporterInput;

import net.sf.jasperreports.export.SimpleOutputStreamExporterOutput;

import net.sf.jasperreports.export.SimplePdfExporterConfiguration;

import net.sf.jasperreports.view.JasperViewer;

public class ReportGenerator {

public void generatevoucher(ObservableList<Sale> sale) throws JRException, FileNotFoundException,

ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

BasicConfigurator.configure();

//

// new DBInitialize().DBInitialize();

// Connection conn = ConnectionUtils.getConnection();

// then using empty datasource.

// JRDataSource dataSource = new JREmptyDataSource();

HashMap<String, Object> param = new HashMap<String, Object>();

/\* List to hold Items \*/

List<Sale> listItems = new ArrayList<Sale>();

for (int i = 0; i < sale.size(); i++) {

Sale sa = new Sale();

sa.setName(sale.get(i).getName());

sa.setUnitamount(sale.get(i).getUnitamount());

sa.setDiscount(sale.get(i).getDiscount());

sa.setQuantity(sale.get(i).getQuantity());

sa.setTotalamount(sale.get(i).getTotalamount());

listItems.add(sa);

}

/\* Convert List to JRBeanCollectionDataSource \*/

JRBeanCollectionDataSource itemsJRBean = new JRBeanCollectionDataSource(listItems);

// current date

String pattern = "dd/MM/yyyy";

String todaydate = new SimpleDateFormat(pattern).format(new Date());

// create today current time(purhcase time)

String hour = "" + new Date().getHours();

String min = "" + new Date().getMinutes();

String time = hour + " : " + min;

System.out.println("purhcase time is : " + time);

/\*

\* //current time DateFormat format = new SimpleDateFormat("HHmm"); String time

\* = format.format(new Date()); String time1 = time.split(1);

\* System.out.println("time is : "+time);

\*/

ObservableList<String> buyGetList = FXCollections.observableArrayList(Common.buygetdata);

ObservableList<String> toPrintPromoList = FXCollections.observableArrayList();

for(int i =0; i < buyGetList.size(); i++) {

//get buy get by promotion

String getPromoQuery = "SELECT productitems.name , promotion.description FROM promotion, productitems WHERE promotion.productid = '"+buyGetList.get(i)+"' AND promotion.productid = productitems.barcode";

String promoDesc = "";

String itemName = "";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsPromoDesc = DBInitialize.statement.executeQuery(getPromoQuery);

while(rsPromoDesc.next()) {

itemName = rsPromoDesc.getString(1);

promoDesc = rsPromoDesc.getString(2);

}

toPrintPromoList.add("For Item : "+ itemName +" : "+promoDesc);

}//ennd of for

//get id of buy get list and check the quantity and reduce to db

for(int e =0 ; e < buyGetList.size(); e++) {

String barcode = buyGetList.get(e);

for(int f = 0 ; f < sale.size() ; f++) {

Sale sale1 = sale.get(e);

if(barcode.equals(sale1.getBarcode())){

int count = sale1.getQuantity();

//get buy get data

String getBuyGetQuery = "SELECT `description` FROM `promotion` WHERE promotion.productid = '"+barcode+"'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsBuyGet = DBInitialize.statement.executeQuery(getBuyGetQuery);

String buyGetData = "";

while(rsBuyGet.next()) {

buyGetData = rsBuyGet.getString(1);

}//end of while

String[] buyGetDataAry = buyGetData.split("\\s+");

String buy = buyGetDataAry[1];

String get = buyGetDataAry[3];

if(count >= Integer.parseInt(buy)) {

int tempGive = (int)(count / Integer.parseInt(buy));

int realGive = (int)(tempGive \* Integer.parseInt(get));

//get current stock amount and count

String getCurrentStockAndCountQuery = "SELECT `stockamount`, `count` FROM `productitems` WHERE productitems.barcode = '"+barcode+"'";

new DBInitialize();

ResultSet rsCSAC = DBInitialize.statement.executeQuery(getCurrentStockAndCountQuery);

String curStock = "";

int curCount = 0;

while(rsCSAC.next()) {

curStock = rsCSAC.getString(1);

curCount = rsCSAC.getInt(2);

}

String newStock = ""+(Integer.parseInt(curStock) - realGive);

int newCount = curCount + realGive;

//reduce the stock amount and count in db

String reductStockACountQuery = "UPDATE `productitems` SET `stockamount`= '"+newStock+"' ,`count`= "+newCount+" WHERE productitems.barcode = '"+barcode+"'";

new DBInitialize();

DBInitialize.statement.executeUpdate(reductStockACountQuery);

}//end of check count if

else {

//do nothing

}

}//end of if

}//end of for

}//end of for

//getItems from toPrintPromoList

String toPrintPromo ="";

for( int j =0 ; j < toPrintPromoList.size() ; j++) {

toPrintPromo = toPrintPromo +"\n"+ toPrintPromoList.get(j);

}

param.put("ItemDataSource", "HELLO");

param.put("DS1", itemsJRBean);

param.put("cashiername", Common.cashierrec.getName());

param.put("total", Common.totalAmount);

param.put("pay", Common.payamount);

param.put("change", Common.change);

param.put("date", todaydate);

param.put("time", time);

param.put("slipno", Common.slipno);

param.put("paidtype", Common.paidtype);

param.put("buygetpromo", toPrintPromo);

param.put("cardinfo", Common.cardinfo);

// Make sure the output directory exists.

File outDir = new File("../Desktop/UCSMPOS");

outDir.mkdirs();

//JasperCompileManager.compileReportToFile(new File("").getAbsolutePath() + "/src/jaspertemplate/voucherprint.jrxml", new File("").getAbsolutePath() + "/src/jaspertemplate/voucherprint.jasper");

//JasperCompileManager.compileReportToFile(new File("").getAbsolutePath() + "/src/jaspertemplate/voucherprint.jrxml", new File("").getAbsolutePath() + "/src/jaspertemplate/voucherprint.jasper");

JasperReport jasperReport = (JasperReport) JRLoader.loadObjectFromFile(new File("").getAbsolutePath() +"/src/jaspertemplate/voucherprint.jasper");

//JasperDesign jasperDesign = JRXmlLoader.load(new File("").getAbsolutePath() + "/src/jaspertemplate/voucherprint.jasper");

//System.out.println("file is : " + new File("").getAbsolutePath() + "/src/jaspertemplate/voucherprint.jasper");

// OutputStream outputfile = new FileOutputStream(new

// File("/Users/tylersai/Desktop/jaspervoucher.pdf"));

//JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, param, new JREmptyDataSource());

// JasperExportManager.exportReportToPdfStream(jasperPrint, outputfile);

// PDF Exportor.

JRPdfExporter exporter = new JRPdfExporter();

ExporterInput exporterInput = new SimpleExporterInput(jasperPrint);

// ExporterInput

exporter.setExporterInput(exporterInput);

// ExporterOutput

OutputStreamExporterOutput exporterOutput = new SimpleOutputStreamExporterOutput(

"../Desktop/UCSMPOS/Voucher.pdf");

// Output

exporter.setExporterOutput(exporterOutput);

//

SimplePdfExporterConfiguration configuration = new SimplePdfExporterConfiguration();

exporter.setConfiguration(configuration);

exporter.exportReport();

JasperViewer jasperViewer = new JasperViewer(jasperPrint, false);

jasperViewer.setVisible(true);

jasperViewer.setFitPageZoomRatio();

jasperViewer.setTitle("M&M's Sale System: Printing service");

// jasperViewer.getIc

Common.buygetdata.clear();

Common.buygetitem = "";

Common.buygetpromo = "";

Common.totalAmount = 0;

Common.change = 0;

Common.payamount = 0;

}

///daily report generation

public void generateDailyReport() throws ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException, JRException {

BasicConfigurator.configure();

// current date

String pattern = "dd/MM/yyyy";

String today = new SimpleDateFormat(pattern).format(new Date());

System.out.println("Today is ....... " + today);

// get today total sale amount

String getTotalSaleQuery = "SELECT `totalamount` FROM `purchase` WHERE purchase.date = '"+today+"'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsGetTotalAmount = DBInitialize.statement.executeQuery(getTotalSaleQuery);

double totalAmount = 0;

while(rsGetTotalAmount.next()) {

totalAmount = totalAmount + Double.parseDouble(rsGetTotalAmount.getString(1));

}

//get today total sale count

String getTotalItemCount = "SELECT `quantity` FROM `purchase` WHERE purchase.date = '"+today+"'";

new DBInitialize();

ResultSet rsGetTotalItemCount = DBInitialize.statement.executeQuery(getTotalItemCount);

int itemCount = 0;

while(rsGetTotalItemCount.next()) {

String tempCount = rsGetTotalItemCount.getString(1);

String[] tempAry = tempCount.split(",");

for( int i = 0 ; i < tempAry.length ; i++) {

System.out.println("count is : "+tempAry[i]);

itemCount = itemCount + Integer.parseInt(tempAry[i]);

}

}

System.out.println("Total sale item count : "+itemCount);

//compute cash and card user for today

//get total purchase

String getTotalPurchaseCountQuery = "SELECT COUNT(\*) FROM `purchase` WHERE purchase.date = '"+today+"'";

int totalPurchase = 0;

new DBInitialize();

ResultSet rsTotalPurchaseCount = DBInitialize.statement.executeQuery(getTotalPurchaseCountQuery);

while(rsTotalPurchaseCount.next()) {

totalPurchase = rsTotalPurchaseCount.getInt(1);

}

//get total card user (from transaction table)

String getCardUserCountQuery = "SELECT COUNT(\*) FROM transaction, purchase WHERE transaction.purchaseid = purchase.id AND purchase.date = '"+today+"'";

int totalCard = 0;

new DBInitialize();

ResultSet rsTotalCardUser = DBInitialize.statement.executeQuery(getCardUserCountQuery);

while(rsTotalCardUser.next()) {

totalCard = rsTotalCardUser.getInt(1);

}

int totalCash = totalPurchase - totalCard ;

//get category from db

//get category list from db

ObservableList<String> categoryList = FXCollections.observableArrayList();

String getCagetoryListQuery = "SELECT `name` FROM `productcategory`;";

new DBInitialize();

ResultSet rsCategory = DBInitialize.statement.executeQuery(getCagetoryListQuery);

while(rsCategory.next()) {

categoryList.add(rsCategory.getString(1));

}

//create array for category name, count , total sale amount

String [] categoryNameAry = new String[categoryList.size()];

int [] categoryCountAry = new int [categoryList.size()];

Double [] categoryTotalAmountAry = new Double[categoryList.size()];

//add item to category name array

for(int a = 0 ; a < categoryNameAry.length; a++) {

categoryNameAry[a] = categoryList.get(a);

categoryCountAry[a] = 0;

categoryTotalAmountAry[a] = 0.0 ;

}

ObservableList<String> barcodeAryList = FXCollections.observableArrayList();

ObservableList<String> qtyAryList = FXCollections.observableArrayList();

//get barcode and find category type

String getPurchaseBarcodeQuery = "SELECT `barcode`, `quantity` FROM `purchase` WHERE purchase.date = '"+today+"'";

new DBInitialize();

ResultSet rsPurchaseBarcode = DBInitialize.statement.executeQuery(getPurchaseBarcodeQuery);

while(rsPurchaseBarcode.next()) {

String barcode = rsPurchaseBarcode.getString(1);

String qty = rsPurchaseBarcode.getString(2);

String[] barcodeAry = barcode.split(",");

String[] qtyAry = qty.split(",");

for(int e = 0 ; e < barcodeAry.length; e++) {

barcodeAryList.add(barcodeAry[e]);

qtyAryList.add(qtyAry[e]);

}

}//end of while

for(int i = 0 ; i < barcodeAryList.size(); i++) {

String getCategoryQuery = "SELECT productcategory.name , productitems.price FROM productcategory, productitems WHERE productitems.barcode = '"+barcodeAryList.get(i)+"' AND productitems.categoryid = productcategory.id;";

new DBInitialize();

ResultSet rsBarcodeToGategory = DBInitialize.statement.executeQuery(getCategoryQuery);

while(rsBarcodeToGategory.next()) {

String categoryName = rsBarcodeToGategory.getString(1);

String price = rsBarcodeToGategory.getString(2);

for(int k = 0 ; k < categoryNameAry.length; k++) {

if(categoryNameAry[k].equals(categoryName)) {

categoryCountAry[k] = categoryCountAry[k] + Integer.parseInt(qtyAryList.get(i)); //set count

categoryTotalAmountAry[k] = categoryTotalAmountAry[k] + Double.parseDouble(price) \* Integer.parseInt(qtyAryList.get(i)) ;

}//end of if

}//end of inner for

}//end of inner while

}//end of for

double initialsale = 0;

double promotion = 0 ;

/\* List to hold Items \*/

List<DailyReport> listItems = new ArrayList<DailyReport>();

//listItems.add(new DailyReport("Sattionary", 4, 3000.0));

for(int f = 0 ; f < categoryNameAry.length; f++) {

DailyReport dr = new DailyReport();

dr.setCategoryname(categoryNameAry[f]);

dr.setSalecount(categoryCountAry[f]);

dr.setCategorytotalamount(categoryTotalAmountAry[f]);

listItems.add(dr);

initialsale = initialsale + categoryTotalAmountAry[f] ;

}

promotion = initialsale - totalAmount;

/////////////////////////////////////////////////////////

//for daily item sale table

//list

ObservableList<String> barcodeList = FXCollections.observableArrayList();

//ObservableList<String> quantityList = FXCollections.observableArrayList();

//query for getting all the item data from db

String getAllItemQuery = "SELECT `barcode` FROM `productitems`";

new DBInitialize();

ResultSet rsAllItem = DBInitialize.statement.executeQuery(getAllItemQuery);

while(rsAllItem.next()) {

barcodeList.add(rsAllItem.getString(1));

}

String[] barcodeAry = new String[barcodeList.size()];

String[] barcodeCountAry = new String[barcodeList.size()];

String[] barcodePriceAry = new String[barcodeList.size()];

String[] barcodeCategoryAry = new String[barcodeList.size()];

String[] barcodeNameAry = new String[barcodeList.size()];

//add data to count

for( int b = 0 ; b < barcodeCountAry.length; b++) {

barcodeCountAry[b] = "0";

barcodeAry[b] = barcodeList.get(b);

}

//barcodeArrayList & qty array list

ObservableList<String> barcodeArrayList = FXCollections.observableArrayList();

ObservableList<String> qtyArrayList = FXCollections.observableArrayList();

//query

String getPurchaseTodayQuery = "SELECT `barcode`, `quantity` FROM `purchase` WHERE purchase.date = '"+today+"'";

new DBInitialize();

ResultSet rsPurchaseToday = DBInitialize.statement.executeQuery(getPurchaseTodayQuery);

while(rsPurchaseToday.next()) {

String barcodeTemp = (rsPurchaseToday.getString(1));

String qtyTodayTemp = rsPurchaseToday.getString(2);

String[] barcodeTempAry = barcodeTemp.split(",");

String[] qtyTodayTempAry = qtyTodayTemp.split(",");

for(int w = 0 ; w < barcodeTempAry.length; w++) {

barcodeArrayList.add(barcodeTempAry[w]);

qtyArrayList.add(qtyTodayTempAry[w]);

}

}//end of while

for(int m =0 ; m < barcodeAryList.size(); m++) {

String bCode = barcodeAryList.get(m);

for(int l = 0; l < barcodeAry.length; l++) {

if(bCode.equals(barcodeAry[l])){

barcodeCountAry[l] = "" + (Integer.parseInt(barcodeCountAry[l]) + Integer.parseInt(qtyArrayList.get(m)) );

}//end of if

}//end of inner for

}//end of for

//get category name and price in ordering

for(int g = 0 ; g < barcodeAry.length; g ++) {

String getCandPQuery = "SELECT productcategory.name , productitems.price, productitems.name FROM productitems, productcategory WHERE productitems.barcode = '"+barcodeAry[g]+"' AND productitems.categoryid = productcategory.id";

new DBInitialize();

ResultSet rsGetCandP = DBInitialize.statement.executeQuery(getCandPQuery);

while(rsGetCandP.next()) {

String cName = rsGetCandP.getString(1);

String cPrice = rsGetCandP.getString(2);

String pName = rsGetCandP.getString(3);

barcodeCategoryAry[g] = cName;

barcodePriceAry[g] = cPrice;

barcodeNameAry[g] = pName;

}//end of while

}//end of for loop

/\*//check point

System.out.println("checking ............");

for(int i =0 ; i< barcodeAry.length ; i++) {

System.out.print(barcodeAry[i]);

System.out.print(barcodeCategoryAry[i]);

System.out.print(barcodeNameAry[i]);

System.out.print(barcodeCountAry[i]);

}\*/

/\* List to hold Items \*/

List<DayItemTable> listItems1 = new ArrayList<DayItemTable>();

//construct object from model

for(int v = 0 ; v < barcodeAry.length; v++) {

DayItemTable dit = new DayItemTable();

dit.setBarcode(barcodeAry[v]);

dit.setCategory(barcodeCategoryAry[v]);

dit.setName(barcodeNameAry[v]);

dit.setSalecount(barcodeCountAry[v]);

Double amount = Integer.parseInt(barcodeCountAry[v]+"") \* Double.parseDouble(barcodePriceAry[v]);

dit.setSaleamount(amount+"");

listItems1.add(dit);

}

/\*--------------------------------------------------\*/

HashMap<String, Object> param = new HashMap<String, Object>();

/\* Convert List to JRBeanCollectionDataSource \*/

JRBeanCollectionDataSource itemsJRBean = new JRBeanCollectionDataSource(listItems);

JRBeanCollectionDataSource itemsJRBean1 = new JRBeanCollectionDataSource(listItems1);

param.put("CategoryDataset", "HELLO");

param.put("ItemDataset", "HELLO");

param.put("DS", itemsJRBean);

param.put("DSItem", itemsJRBean1);

param.put("totalsaleamount", Double.parseDouble(""+totalAmount));

param.put("totalsaleitemcount", Integer.parseInt(""+itemCount));

param.put("cardcustomer", Integer.parseInt(""+totalCard));

param.put("cashpaidcustomer", Integer.parseInt(""+totalCash));

param.put("initialsale", ""+initialsale);

param.put("netsale", ""+totalAmount);

param.put("promotion",""+promotion );

// Make sure the output directory exists.

File outDir = new File("../Desktop/UCSMPOS");

outDir.mkdirs();

JasperReport jasperReport = (JasperReport) JRLoader.loadObjectFromFile(new File("").getAbsolutePath() +"/src/jaspertemplate/daily\_report.jasper");

//JasperDesign jasperDesign = JRXmlLoader.load(new File("").getAbsolutePath() + "/src/jaspertemplate/daily\_report.jasper");

System.out.println("file is : " + new File("").getAbsolutePath() + "/src/jaspertemplate/daily\_report.jasper");

/\* Using compiled version(.jasper) of Jasper report to generate PDF \*/

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, param, new JREmptyDataSource());

// OutputStream outputfile = new FileOutputStream(new

// File("/Users/tylersai/Desktop/jaspervoucher.pdf"));

//JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);

//JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, param, new JREmptyDataSource());

// JasperExportManager.exportReportToPdfStream(jasperPrint, outputfile);

// PDF Exportor.

JRPdfExporter exporter = new JRPdfExporter();

ExporterInput exporterInput = new SimpleExporterInput(jasperPrint);

// ExporterInput

exporter.setExporterInput(exporterInput);

// ExporterOutput

OutputStreamExporterOutput exporterOutput = new SimpleOutputStreamExporterOutput(

"../Desktop/UCSMPOS/daily\_report.pdf");

// Output

exporter.setExporterOutput(exporterOutput);

//

SimplePdfExporterConfiguration configuration = new SimplePdfExporterConfiguration();

exporter.setConfiguration(configuration);

exporter.exportReport();

JasperViewer jasperViewer = new JasperViewer(jasperPrint, false);

jasperViewer.setVisible(true);

jasperViewer.setFitPageZoomRatio();

jasperViewer.setTitle("M&M's Sale System: Printing service");

// jasperViewer.getIc

}

public void generatePopularItem() throws JRException, ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

BasicConfigurator.configure();

ObservableList<ProductItem> popularData = FXCollections.observableArrayList();

//get popular item data from db

new DBInitialize().DBInitialize();

String query = "SELECT productitems.barcode, productitems.name, productcategory.name, productitems.price, supplier.companyname, productitems.dateadded, productitems.stockamount, productitems.expireddate, productitems.count FROM productitems, supplier,productcategory WHERE productitems.categoryid = productcategory.id AND productitems.supplierid = supplier.id ORDER BY productitems.count DESC LIMIT 25";

new DBInitialize();

ResultSet rs = DBInitialize.statement.executeQuery(query);

while(rs.next()) {

ProductItem product = new ProductItem();

product.setBarcode(rs.getString(1));

product.setName(rs.getString(2));

product.setCategoryname(rs.getString(3));

product.setPrice(rs.getString(4));

product.setSuppliername(rs.getString(5));

product.setDateadded(rs.getString(6));

product.setStockamount(rs.getString(7));

product.setExpiredate(rs.getString(8));

product.setCount(rs.getInt(9));

popularData.add(product);

}

HashMap<String, Object> param = new HashMap<String, Object>();

/\* List to hold Items

List<Sale> listItems = new ArrayList<Sale>();\*/

/\* Convert List to JRBeanCollectionDataSource \*/

JRBeanCollectionDataSource itemsJRBean = new JRBeanCollectionDataSource(popularData);

param.put("PopularDataset", "HELLO");

param.put("DS1", itemsJRBean);

// Make sure the output directory exists.

File outDir = new File("../Desktop/UCSMPOS");

outDir.mkdirs();

//JasperDesign jasperDesign = JRXmlLoader.load(new File("").getAbsolutePath() + "/src/jaspertemplate/popular\_item\_report.jasper");

// System.out.println("file is : " + new File("").getAbsolutePath() + "/src/jaspertemplate/popular\_item\_report.jasper");

JasperReport jasperReport = (JasperReport) JRLoader.loadObjectFromFile(new File("").getAbsolutePath() +"/src/jaspertemplate/popular\_item\_report.jasper");

// OutputStream outputfile = new FileOutputStream(new

// File("/Users/tylersai/Desktop/jaspervoucher.pdf"));

//JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, param, new JREmptyDataSource());

// JasperExportManager.exportReportToPdfStream(jasperPrint, outputfile);

// PDF Exportor.

JRPdfExporter exporter = new JRPdfExporter();

ExporterInput exporterInput = new SimpleExporterInput(jasperPrint);

// ExporterInput

exporter.setExporterInput(exporterInput);

// ExporterOutput

OutputStreamExporterOutput exporterOutput = new SimpleOutputStreamExporterOutput(

"../Desktop/UCSMPOS/Popular\_item.pdf");

// Output

exporter.setExporterOutput(exporterOutput);

//

SimplePdfExporterConfiguration configuration = new SimplePdfExporterConfiguration();

exporter.setConfiguration(configuration);

exporter.exportReport();

JasperViewer jasperViewer = new JasperViewer(jasperPrint, false);

jasperViewer.setVisible(true);

jasperViewer.setFitPageZoomRatio();

jasperViewer.setTitle("M&M's Sale System: Printing service");

}

/\*-----------------------------------------------------------------------------------------------\*/

public void generateMonthlyReport() throws JRException, ClassNotFoundException, InstantiationException, IllegalAccessException, SQLException {

BasicConfigurator.configure();

// current date

String pattern = "dd/MM/yyyy";

String todayy = new SimpleDateFormat(pattern).format(new Date());

System.out.println("Today is ....... " + todayy);

//get month

String [] todayAry = todayy.split("/");

String currentMonth = todayAry[1] + "/" + todayAry[2];

// get today total sale amount

String getTotalSaleQuery = "SELECT `totalamount` FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

new DBInitialize().DBInitialize();

new DBInitialize();

ResultSet rsGetTotalAmount = DBInitialize.statement.executeQuery(getTotalSaleQuery);

double totalAmount = 0;

while(rsGetTotalAmount.next()) {

totalAmount = totalAmount + Double.parseDouble(rsGetTotalAmount.getString(1));

}

//get today total sale count

String getTotalItemCount = "SELECT `quantity` FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

new DBInitialize();

ResultSet rsGetTotalItemCount = DBInitialize.statement.executeQuery(getTotalItemCount);

int itemCount = 0;

while(rsGetTotalItemCount.next()) {

String tempCount = rsGetTotalItemCount.getString(1);

String[] tempAry = tempCount.split(",");

for( int i = 0 ; i < tempAry.length ; i++) {

System.out.println("count is : "+tempAry[i]);

itemCount = itemCount + Integer.parseInt(tempAry[i]);

}

}

System.out.println("Total sale item count : "+itemCount);

//compute cash and card user for today

//get total purchase

String getTotalPurchaseCountQuery = "SELECT COUNT(\*) FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

int totalPurchase = 0;

new DBInitialize();

ResultSet rsTotalPurchaseCount = DBInitialize.statement.executeQuery(getTotalPurchaseCountQuery);

while(rsTotalPurchaseCount.next()) {

totalPurchase = rsTotalPurchaseCount.getInt(1);

}

//get total card user (from transaction table)

String getCardUserCountQuery = "SELECT COUNT(\*) FROM transaction, purchase WHERE transaction.purchaseid = purchase.id AND purchase.date LIKE '%"+currentMonth+"'";

int totalCard = 0;

new DBInitialize();

ResultSet rsTotalCardUser = DBInitialize.statement.executeQuery(getCardUserCountQuery);

while(rsTotalCardUser.next()) {

totalCard = rsTotalCardUser.getInt(1);

}

int totalCash = totalPurchase - totalCard ;

//get category from db

//get category list from db

ObservableList<String> categoryList = FXCollections.observableArrayList();

String getCagetoryListQuery = "SELECT `name` FROM `productcategory`;";

new DBInitialize();

ResultSet rsCategory = DBInitialize.statement.executeQuery(getCagetoryListQuery);

while(rsCategory.next()) {

categoryList.add(rsCategory.getString(1));

}

//create array for category name, count , total sale amount

String [] categoryNameAry = new String[categoryList.size()];

int [] categoryCountAry = new int [categoryList.size()];

Double [] categoryTotalAmountAry = new Double[categoryList.size()];

//add item to category name array

for(int a = 0 ; a < categoryNameAry.length; a++) {

categoryNameAry[a] = categoryList.get(a);

categoryCountAry[a] = 0;

categoryTotalAmountAry[a] = 0.0 ;

}

ObservableList<String> barcodeAryList = FXCollections.observableArrayList();

ObservableList<String> qtyAryList = FXCollections.observableArrayList();

//get barcode and find category type

String getPurchaseBarcodeQuery = "SELECT `barcode`, `quantity` FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

new DBInitialize();

ResultSet rsPurchaseBarcode = DBInitialize.statement.executeQuery(getPurchaseBarcodeQuery);

while(rsPurchaseBarcode.next()) {

String barcode = rsPurchaseBarcode.getString(1);

String qty = rsPurchaseBarcode.getString(2);

String[] barcodeAry = barcode.split(",");

String[] qtyAry = qty.split(",");

for(int e = 0 ; e < barcodeAry.length; e++) {

barcodeAryList.add(barcodeAry[e]);

qtyAryList.add(qtyAry[e]);

}

}//end of while

for(int i = 0 ; i < barcodeAryList.size(); i++) {

String getCategoryQuery = "SELECT productcategory.name , productitems.price FROM productcategory, productitems WHERE productitems.barcode = '"+barcodeAryList.get(i)+"' AND productitems.categoryid = productcategory.id;";

new DBInitialize();

ResultSet rsBarcodeToGategory = DBInitialize.statement.executeQuery(getCategoryQuery);

while(rsBarcodeToGategory.next()) {

String categoryName = rsBarcodeToGategory.getString(1);

String price = rsBarcodeToGategory.getString(2);

for(int k = 0 ; k < categoryNameAry.length; k++) {

if(categoryNameAry[k].equals(categoryName)) {

categoryCountAry[k] = categoryCountAry[k] + Integer.parseInt(qtyAryList.get(i)); //set count

categoryTotalAmountAry[k] = categoryTotalAmountAry[k] + Double.parseDouble(price) \* Integer.parseInt(qtyAryList.get(i)) ;

}//end of if

}//end of inner for

}//end of inner while

}//end of for

double initialsale = 0;

double promotion = 0 ;

/\* List to hold Items \*/

List<DailyReport> listItems = new ArrayList<DailyReport>();

//listItems.add(new DailyReport("Sattionary", 4, 3000.0));

for(int f = 0 ; f < categoryNameAry.length; f++) {

DailyReport dr = new DailyReport();

dr.setCategoryname(categoryNameAry[f]);

dr.setSalecount(categoryCountAry[f]);

dr.setCategorytotalamount(categoryTotalAmountAry[f]);

listItems.add(dr);

initialsale = initialsale + categoryTotalAmountAry[f] ;

}

promotion = initialsale - totalAmount;

/////////////////////////////////////////////////////////

//for daily item sale table

//list

ObservableList<String> barcodeList = FXCollections.observableArrayList();

//ObservableList<String> quantityList = FXCollections.observableArrayList();

//query for getting all the item data from db

String getAllItemQuery = "SELECT `barcode` FROM `productitems`";

new DBInitialize();

ResultSet rsAllItem = DBInitialize.statement.executeQuery(getAllItemQuery);

while(rsAllItem.next()) {

barcodeList.add(rsAllItem.getString(1));

}

String[] barcodeAry = new String[barcodeList.size()];

String[] barcodeCountAry = new String[barcodeList.size()];

String[] barcodePriceAry = new String[barcodeList.size()];

String[] barcodeCategoryAry = new String[barcodeList.size()];

String[] barcodeNameAry = new String[barcodeList.size()];

//add data to count

for( int b = 0 ; b < barcodeCountAry.length; b++) {

barcodeCountAry[b] = "0";

barcodeAry[b] = barcodeList.get(b);

}

//barcodeArrayList & qty array list

ObservableList<String> barcodeArrayList = FXCollections.observableArrayList();

ObservableList<String> qtyArrayList = FXCollections.observableArrayList();

//query

String getPurchaseTodayQuery = "SELECT `barcode`, `quantity` FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

new DBInitialize();

ResultSet rsPurchaseToday = DBInitialize.statement.executeQuery(getPurchaseTodayQuery);

while(rsPurchaseToday.next()) {

String barcodeTemp = (rsPurchaseToday.getString(1));

String qtyTodayTemp = rsPurchaseToday.getString(2);

String[] barcodeTempAry = barcodeTemp.split(",");

String[] qtyTodayTempAry = qtyTodayTemp.split(",");

for(int w = 0 ; w < barcodeTempAry.length; w++) {

barcodeArrayList.add(barcodeTempAry[w]);

qtyArrayList.add(qtyTodayTempAry[w]);

}

}//end of while

for(int m =0 ; m < barcodeAryList.size(); m++) {

String bCode = barcodeAryList.get(m);

for(int l = 0; l < barcodeAry.length; l++) {

if(bCode.equals(barcodeAry[l])){

barcodeCountAry[l] = "" + (Integer.parseInt(barcodeCountAry[l]) + Integer.parseInt(qtyArrayList.get(m)) );

}//end of if

}//end of inner for

}//end of for

//get category name and price in ordering

for(int g = 0 ; g < barcodeAry.length; g ++) {

String getCandPQuery = "SELECT productcategory.name , productitems.price, productitems.name FROM productitems, productcategory WHERE productitems.barcode = '"+barcodeAry[g]+"' AND productitems.categoryid = productcategory.id";

new DBInitialize();

ResultSet rsGetCandP = DBInitialize.statement.executeQuery(getCandPQuery);

while(rsGetCandP.next()) {

String cName = rsGetCandP.getString(1);

String cPrice = rsGetCandP.getString(2);

String pName = rsGetCandP.getString(3);

barcodeCategoryAry[g] = cName;

barcodePriceAry[g] = cPrice;

barcodeNameAry[g] = pName;

}//end of while

}//end of for loop

//check point

System.out.println("checking ............");

for(int i =0 ; i< barcodeAry.length ; i++) {

System.out.print(barcodeAry[i]);

System.out.print(barcodeCategoryAry[i]);

System.out.print(barcodeNameAry[i]);

System.out.print(barcodeCountAry[i]);

}

/\* List to hold Items \*/

List<DayItemTable> listItems1 = new ArrayList<DayItemTable>();

//construct object from model

for(int v = 0 ; v < barcodeAry.length; v++) {

DayItemTable dit = new DayItemTable();

dit.setBarcode(barcodeAry[v]);

dit.setCategory(barcodeCategoryAry[v]);

dit.setName(barcodeNameAry[v]);

dit.setSalecount(barcodeCountAry[v]);

Double amount = Integer.parseInt(barcodeCountAry[v]+"") \* Double.parseDouble(barcodePriceAry[v]);

dit.setSaleamount(amount+"");

listItems1.add(dit);

}

/\*--------------------------------------------------\*/

//for daily sale items

//days array

String[] daysAry = new String[31];

String[] daysAmountAry = new String[31];

String[] daysCountAry = new String[31];

//set data to arrays

for(int z =0 ; z < daysAry.length; z++) {

daysAry[z] = String.format("%02d", (z+1));

daysAmountAry[z] = 0+"";

daysCountAry[z] = "0";

}

//get daily purchase in this month

String getDailyPurchaseQuery = "SELECT `date`, `totalamount` FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

new DBInitialize();

ResultSet rsDailyPurchase = DBInitialize.statement.executeQuery(getDailyPurchaseQuery);

while(rsDailyPurchase.next()) {

String dateTemp = rsDailyPurchase.getString(1);

String totalAmountTemp = rsDailyPurchase.getString(2);

String[] dateTempAry = dateTemp.split("/");

//String totalAmountTempAry

for(int r =0 ; r < daysAry.length; r++) {

if(dateTempAry[0].equals(daysAry[r])) {

daysAmountAry[r] = "" + (Double.parseDouble(daysAmountAry[r]) + Double.parseDouble(totalAmountTemp));

}

}

}

//get the total count , initial sale and discount

for(int s =0 ; s < daysAry.length; s++) {

//get totalCount

String getTotalCountQueryy = "SELECT `quantity` FROM `purchase` WHERE purchase.date = '"+daysAry[s]+"/"+currentMonth+"'";

new DBInitialize();

ResultSet rsTotalCountt = DBInitialize.statement.executeQuery(getTotalCountQueryy);

while(rsTotalCountt.next() ) {

String coulntTemppp = rsTotalCountt.getString(1);

String[] countTempAryyy = coulntTemppp.split(",");

for(int c =0 ; c < countTempAryyy.length; c++) {

daysCountAry[s] = "" + ( Integer.parseInt(daysCountAry[s]) + Integer.parseInt(countTempAryyy[c]));

}//end of inner for

}//end of while

/\*//get price and compute initial sale

String getDailyBarcodeQuery = "SELECT `date`, `barcode` FROM `purchase` WHERE purchase.date LIKE '%"+currentMonth+"'";

new DBInitialize();

ResultSet rsDailyBarcode = DBInitialize.statement.executeQuery(getDailyBarcodeQuery);

while(rsDailyBarcode.next()) {

String dailyDate = rsDailyBarcode.getString(1);

String Dailybarcode = rsDailyBarcode.getString(2);

} \*/

}//end of for

/\* List to hold Items \*/

List<DailyTable> listItems2 = new ArrayList<DailyTable>();

//adding data object

for(int q =0; q < daysAry.length; q ++) {

DailyTable dat = new DailyTable();

dat.setDate(daysAry[q]+"/"+currentMonth);

dat.setItemcount(daysCountAry[q]);

dat.setNetsale(daysAmountAry[q]);

listItems2.add(dat);

}

/\*--------------------------------------------------\*/

HashMap<String, Object> param = new HashMap<String, Object>();

/\* Convert List to JRBeanCollectionDataSource \*/

JRBeanCollectionDataSource itemsJRBean = new JRBeanCollectionDataSource(listItems);

JRBeanCollectionDataSource itemsJRBean1 = new JRBeanCollectionDataSource(listItems1);

JRBeanCollectionDataSource itemsJRBean2 = new JRBeanCollectionDataSource(listItems2);

param.put("CategoryDataset", "HELLO");

param.put("ItemDataset", "HELLO");

param.put("Dataset1", "HELLO");

param.put("DS", itemsJRBean);

param.put("DSItem", itemsJRBean1);

param.put("DS1", itemsJRBean2);

param.put("totalsaleamount", Double.parseDouble(""+totalAmount));

param.put("totalsaleitemcount", Integer.parseInt(""+itemCount));

param.put("cardcustomer", Integer.parseInt(""+totalCard));

param.put("cashpaidcustomer", Integer.parseInt(""+totalCash));

param.put("initialsale", ""+initialsale);

param.put("netsale", ""+totalAmount);

param.put("promotion",""+promotion );

param.put("categorysale","Item Sale" );

// Make sure the output directory exists.

File outDir = new File("../Desktop/UCSMPOS");

outDir.mkdirs();

JasperReport jasperReport = (JasperReport) JRLoader.loadObjectFromFile(new File("").getAbsolutePath() +"/src/jaspertemplate/monthly\_report.jasper");

//JasperDesign jasperDesign = JRXmlLoader.load(new File("").getAbsolutePath() + "/src/jaspertemplate/monthly\_report.jasper");

// System.out.println("file is : " + new File("").getAbsolutePath() + "/src/jaspertemplate/monthly\_report.jasper");

// OutputStream outputfile = new FileOutputStream(new

// File("/Users/tylersai/Desktop/jaspervoucher.pdf"));

//JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);

JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, param, new JREmptyDataSource());

// JasperExportManager.exportReportToPdfStream(jasperPrint, outputfile);

// PDF Exportor.

JRPdfExporter exporter = new JRPdfExporter();

ExporterInput exporterInput = new SimpleExporterInput(jasperPrint);

// ExporterInput

exporter.setExporterInput(exporterInput);

// ExporterOutput

OutputStreamExporterOutput exporterOutput = new SimpleOutputStreamExporterOutput(

"../Desktop/UCSMPOS/monthly\_report.pdf");

// Output

exporter.setExporterOutput(exporterOutput);

//

SimplePdfExporterConfiguration configuration = new SimplePdfExporterConfiguration();

exporter.setConfiguration(configuration);

exporter.exportReport();

JasperViewer jasperViewer = new JasperViewer(jasperPrint, false);

jasperViewer.setVisible(true);

jasperViewer.setFitPageZoomRatio();

jasperViewer.setTitle("M&M's Sale System: Printing service");

}

}

**major**

**Login.java**

package major;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.image.Image;

import javafx.stage.Stage;

public class Login extends Application{

//public static Login loginTran =new Login();

public void start(Stage primaryStage) throws Exception {

// TODO Auto-generated method stub

FXMLLoader root = new FXMLLoader(getClass().getResource("/ui/Page\_login.fxml"));

// LoginController logincontroller = root.getController();

Scene scene = new Scene(((Parent) root.load()), 1320,700);

primaryStage.setScene(scene);

primaryStage.setResizable(false);

primaryStage.setMaximized(false);

primaryStage.setTitle("M&M's Sale System");

primaryStage.getIcons().add(new Image("graphic/poslogorect.png"));

primaryStage.show();

}

public static void main(String args[]) {

Application.launch(args);

}

}

**MainAdmin.java**

package major;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class MainAdmin extends Application{

// public static MainCashier mainCashierTran =new MainCashier();

public void start(Stage primaryStage) throws Exception {

// TODO Auto-generated method stub

FXMLLoader root = new FXMLLoader(getClass().getResource("/ui/template/Admin\_main.fxml"));

// LoginController logincontroller = root.getController();

Scene scene = new Scene(((Parent) root.load()),850,500);

primaryStage.setScene(scene);

primaryStage.setResizable(true);

primaryStage.setMaximized(true);

primaryStage.show();

}

public static void main(String args[]) {

Application.launch(args);

}

}

**MainCashier.java**

package major;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class MainCashier extends Application{

// public static MainCashier mainCashierTran =new MainCashier();

public void start(Stage primaryStage) throws Exception {

// TODO Auto-generated method stub

FXMLLoader root = new FXMLLoader(getClass().getResource("/ui/template/cashier\_main.fxml"));

// LoginController logincontroller = root.getController();

Scene scene = new Scene(((Parent) root.load()),850,500);

primaryStage.setScene(scene);

primaryStage.setResizable(true);

primaryStage.setMaximized(true);

primaryStage.show();

}

public static void main(String args[]) {

Application.launch(args);

}

}

**model**

**CardUser.java**

package model;

public class CardUser {

String customrid ;

String name;

String age;

String gender;

String address;

String phone;

String email;

String pin;

String amount;

String cardno;

String lastdateused;

String expireddate;

String registerdate;

public CardUser(String customrid, String name, String age, String gender, String address, String phone,

String email, String pin, String amount, String cardno, String lastdateused, String expireddate,

String registerdate) {

super();

this.customrid = customrid;

this.name = name;

this.age = age;

this.gender = gender;

this.address = address;

this.phone = phone;

this.email = email;

this.pin = pin;

this.amount = amount;

this.cardno = cardno;

this.lastdateused = lastdateused;

this.expireddate = expireddate;

this.registerdate = registerdate;

}

public CardUser() {

}

public String getCustomrid() {

return customrid;

}

public void setCustomrid(String customrid) {

this.customrid = customrid;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAge() {

return age;

}

public void setAge(String age) {

this.age = age;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPin() {

return pin;

}

public void setPin(String pin) {

this.pin = pin;

}

public String getAmount() {

return amount;

}

public void setAmount(String amount) {

this.amount = amount;

}

public String getCardno() {

return cardno;

}

public void setCardno(String cardno) {

this.cardno = cardno;

}

public String getLastdateused() {

return lastdateused;

}

public void setLastdateused(String lastdateused) {

this.lastdateused = lastdateused;

}

public String getExpireddate() {

return expireddate;

}

public void setExpireddate(String expireddate) {

this.expireddate = expireddate;

}

public String getRegisterdate() {

return registerdate;

}

public void setRegisterdate(String registerdate) {

this.registerdate = registerdate;

}

}

**Cashier.java**

package model;

public class Cashier {

private String id;

private String name;

private String age;

private String gender;

private String address;

private String phone;

private String email;

private String password;

private String dateCreated;

public Cashier() {

}

public Cashier(String id, String name, String age, String gender, String address, String phone, String email,

String password, String dateCreated) {

super();

this.id = id;

this.name = name;

this.age = age;

this.gender = gender;

this.address = address;

this.phone = phone;

this.email = email;

this.password = password;

this.dateCreated = dateCreated;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAge() {

return age;

}

public void setAge(String string) {

this.age = string;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getDateCreated() {

return dateCreated;

}

public void setDateCreated(String dateCreated) {

this.dateCreated = dateCreated;

}

}

**Category.java**

package model;

public class Category {

String id;

String name;

String dateCreated;

public Category(String id, String name, String dateCreated) {

super();

this.id = id;

this.name = name;

this.dateCreated = dateCreated;

}

public Category() {

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDateCreated() {

return dateCreated;

}

public void setDateCreated(String dateCreated) {

this.dateCreated = dateCreated;

}

}

**DailyReport.java**

package model;

public class DailyReport {

String categoryname;

Integer salecount;

Double categorytotalamount;

public String getCategoryname() {

return categoryname;

}

public void setCategoryname(String categoryname) {

this.categoryname = categoryname;

}

public Integer getSalecount() {

return salecount;

}

public void setSalecount(Integer salecount) {

this.salecount = salecount;

}

public Double getCategorytotalamount() {

return categorytotalamount;

}

public void setCategorytotalamount(Double categorytotalamount) {

this.categorytotalamount = categorytotalamount;

}

public DailyReport(String categoryname, Integer salecount, Double categorytotalamount) {

super();

this.categoryname = categoryname;

this.salecount = salecount;

this.categorytotalamount = categorytotalamount;

}

public DailyReport() {

}

}

**DailyTable.java**

package model;

public class DailyTable {

String date;

String itemcount;

String netsale;

public DailyTable(String date, String itemcount, String netsale) {

super();

this.date = date;

this.itemcount = itemcount;

this.netsale = netsale;

}

public DailyTable() {

super();

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getItemcount() {

return itemcount;

}

public void setItemcount(String itemcount) {

this.itemcount = itemcount;

}

public String getNetsale() {

return netsale;

}

public void setNetsale(String netsale) {

this.netsale = netsale;

}

}

**DayItemTable.java**

package model;

public class DayItemTable {

String barcode;

String name;

String category;

String salecount;

String saleamount;

public DayItemTable(String barcode, String name, String category, String salecount, String saleamount) {

super();

this.barcode = barcode;

this.name = name;

this.category = category;

this.salecount = salecount;

this.saleamount = saleamount;

}

public DayItemTable() {

super();

}

public String getBarcode() {

return barcode;

}

public void setBarcode(String barcode) {

this.barcode = barcode;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getSalecount() {

return salecount;

}

public void setSalecount(String salecount) {

this.salecount = salecount;

}

public String getSaleamount() {

return saleamount;

}

public void setSaleamount(String saleamount) {

this.saleamount = saleamount;

}

}

**ProductItem.java**

package model;

public class ProductItem {

String name;

String categoryid;

String categoryname;

String dateadded;

String expiredate;

String price;

String barcode;

String supplierid;

String suppliername;

String stockamount;

String discount;

String discountmore;

int count ;

public ProductItem(String name, String categoryid, String categoryname, String dateadded, String expiredate,

String price, String barcode, String supplierid, String suppliername, String stockamount) {

super();

this.name = name;

this.categoryid = categoryid;

this.categoryname = categoryname;

this.dateadded = dateadded;

this.expiredate = expiredate;

this.price = price;

this.barcode = barcode;

this.supplierid = supplierid;

this.suppliername = suppliername;

this.stockamount = stockamount;

}

public ProductItem(String name, String categoryid, String dateadded, String expiredate,

String price, String barcode, String supplierid, String stockamount) {

super();

this.name = name;

this.categoryid = categoryid;

this.categoryname = categoryname;

this.dateadded = dateadded;

this.expiredate = expiredate;

this.price = price;

this.barcode = barcode;

this.supplierid = supplierid;

this.suppliername = suppliername;

this.stockamount = stockamount;

}

public String getDiscountmore() {

return discountmore;

}

public void setDiscountmore(String discountmore) {

this.discountmore = discountmore;

}

public String getDiscount() {

return discount;

}

public void setDiscount(String discount) {

this.discount = discount;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public ProductItem() {

}

public String getCategoryname() {

return categoryname;

}

public void setCategoryname(String categoryname) {

this.categoryname = categoryname;

}

public String getSuppliername() {

return suppliername;

}

public void setSuppliername(String suppliername) {

this.suppliername = suppliername;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCategoryid() {

return categoryid;

}

public void setCategoryid(String categoryid) {

this.categoryid = categoryid;

}

public String getDateadded() {

return dateadded;

}

public void setDateadded(String dateadded) {

this.dateadded = dateadded;

}

public String getExpiredate() {

return expiredate;

}

public void setExpiredate(String expiredate) {

this.expiredate = expiredate;

}

public String getPrice() {

return price;

}

public void setPrice(String price) {

this.price = price;

}

public String getBarcode() {

return barcode;

}

public void setBarcode(String barcode) {

this.barcode = barcode;

}

public String getSupplierid() {

return supplierid;

}

public void setSupplierid(String supplierid) {

this.supplierid = supplierid;

}

public String getStockamount() {

return stockamount;

}

public void setStockamount(String stockamount) {

this.stockamount = stockamount;

}

}

**Promotion.java**

package model;

public class Promotion {

String id;

String name;

String productId;

String percentage;

String productName;

String more;

public Promotion(String id, String name, String productId,String productName, String percentage, String more) {

super();

this.id = id;

this.name = name;

this.productId = productId;

this.productName = productName;

this.percentage = percentage;

this.more = more;

}

public Promotion() {

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getProductId() {

return productId;

}

public void setProductId(String productId) {

this.productId = productId;

}

public String getPercentage() {

return percentage;

}

public void setPercentage(String percentage) {

this.percentage = percentage;

}

public String getMore() {

return more;

}

public void setMore(String more) {

this.more = more;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

}

**Purchase.java**

package model;

public class Purchase {

String id;

String cashierid;

String date;

String time;

String barcode;

String quantity;

String totalAmount;

String desc;

public Purchase() {

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getCashierid() {

return cashierid;

}

public void setCashierid(String cashierid) {

this.cashierid = cashierid;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getTime() {

return time;

}

public void setTime(String time) {

this.time = time;

}

public String getBarcode() {

return barcode;

}

public void setBarcode(String barcode) {

this.barcode = barcode;

}

public String getQuantity() {

return quantity;

}

public void setQuantity(String quantity) {

this.quantity = quantity;

}

public String getTotalAmount() {

return totalAmount;

}

public void setTotalAmount(String totalAmount) {

this.totalAmount = totalAmount;

}

public String getDesc() {

return desc;

}

public void setDesc(String desc) {

this.desc = desc;

}

}

**Sale.java**

package model;

public class Sale {

private String barcode;

private String name;

private double unitamount;

private int quantity;

private double discount;

private String discountmore;

private double totalamount;

private String more;

public Sale(String barcode, String name, double unitamount, int quantity, double discount, double totalamount) {

super();

this.barcode = barcode;

this.name = name;

this.unitamount = unitamount;

this.quantity = quantity;

this.discount = discount;

this.totalamount = totalamount;

}

public Sale() {

}

public Sale(String name, double unitamount, int quantity, double totalamount) {

super();

this.name = name;

this.unitamount = unitamount;

this.quantity = quantity;

this.totalamount = totalamount;

}

public Sale(double totalamount) {

}

public String getDiscountmore() {

return discountmore;

}

public void setDiscountmore(String discountmore) {

this.discountmore = discountmore;

}

public String getMore() {

return more;

}

public void setMore(String more) {

this.more = more;

}

public String getBarcode() {

return barcode;

}

public void setBarcode(String barcode) {

this.barcode = barcode;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getUnitamount() {

return unitamount;

}

public void setUnitamount(double unitamount) {

this.unitamount = unitamount;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public double getDiscount() {

return discount;

}

public void setDiscount(double discount) {

this.discount = discount;

}

public double getTotalamount() {

return totalamount;

}

public void setTotalamount(double totalamount) {

this.totalamount = totalamount;

}

}

**Supplier.java**

package model;

public class Supplier {

String id;

String name;

String lastSupplied;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLastSupplied() {

return lastSupplied;

}

public void setLastSupplied(String lastSupplied) {

this.lastSupplied = lastSupplied;

}

public Supplier(String id, String name, String lastSupplied) {

super();

this.id = id;

this.name = name;

this.lastSupplied = lastSupplied;

}

public Supplier() {

}

**user interface UI**

**Admin\_card.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminCardController">

<children>

<TableView fx:id="tb\_card" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="580.0" layoutY="1.0" style="-fx-font-size: 30;" text="Manage Card" />

<Label layoutX="12.0" layoutY="240.0" text="Card Number" />

<Label layoutX="12.0" layoutY="300.0" text="PIN" />

<JFXTextField fx:id="tf\_card\_no" layoutX="100.0" layoutY="233.0" prefHeight="27.0" prefWidth="215.0" promptText="777 777 xxx xxx" />

<JFXTextField fx:id="tf\_pin" layoutX="100.0" layoutY="289.0" prefHeight="27.0" prefWidth="215.0" promptText="1234" />

<JFXButton fx:id="bt\_new" layoutX="100.0" layoutY="400.0" onAction="#onNewAction" prefHeight="27.0" prefWidth="66.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

<JFXButton fx:id="bt\_add" layoutX="200.0" layoutY="400.0" onAction="#onAddAction" prefHeight="27.0" prefWidth="66.0" style="-fx-background-color: #615158;" text="Add" textFill="WHITE" />

</children>

</AnchorPane>

**Admin\_cashier.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXRadioButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.ToggleGroup?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminCashierController">

<children>

<TableView fx:id="tb\_cashier" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="31.0" layoutY="158.0" text="ID" />

<Label layoutX="31.0" layoutY="199.0" text="Name" />

<Label layoutX="31.0" layoutY="240.0" text="Age" />

<Label layoutX="31.0" layoutY="283.0" text="Gender" />

<Label layoutX="31.0" layoutY="327.0" text="Address" />

<Label layoutX="31.0" layoutY="368.0" text="Phone" />

<Label layoutX="31.0" layoutY="412.0" text="Email" />

<Label layoutX="31.0" layoutY="458.0" text="Password" />

<Label layoutX="580.0" layoutY="1.0" style="-fx-font-size: 30;" text="Manage Cashier" />

<JFXTextField fx:id="tf\_id" layoutX="115.0" layoutY="145.0" prefHeight="27.0" prefWidth="197.0" promptText="11001" />

<JFXTextField fx:id="tf\_name" layoutX="115.0" layoutY="187.0" prefHeight="27.0" prefWidth="197.0" promptText="Abdelheq" />

<JFXTextField fx:id="tf\_age" layoutX="115.0" layoutY="230.0" prefHeight="27.0" prefWidth="197.0" promptText="21" />

<JFXRadioButton fx:id="rdo\_male" layoutX="115.0" layoutY="283.0" selected="true" text="Male">

<toggleGroup>

<ToggleGroup fx:id="gender" />

</toggleGroup></JFXRadioButton>

<JFXRadioButton fx:id="rdo\_female" layoutX="190.0" layoutY="283.0" text="Female" toggleGroup="$gender" />

<JFXTextField fx:id="tf\_addr" layoutX="115.0" layoutY="313.0" prefHeight="27.0" prefWidth="197.0" promptText="Batna" />

<JFXTextField fx:id="tf\_phone" layoutX="115.0" layoutY="355.0" prefHeight="27.0" prefWidth="197.0" promptText="0667304629" />

<JFXTextField fx:id="tf\_email" layoutX="115.0" layoutY="399.0" prefHeight="27.0" prefWidth="197.0" promptText="example@mail.com" />

<JFXTextField fx:id="tf\_password" layoutX="115.0" layoutY="444.0" prefHeight="27.0" prefWidth="197.0" promptText="cashier" />

<Label layoutX="20.0" layoutY="500.0" text="Date Created" />

<JFXTextField fx:id="tf\_date\_created" layoutX="115.0" layoutY="488.0" prefHeight="27.0" prefWidth="197.0" promptText="00/00/0000" />

<JFXButton fx:id="bt\_new" layoutX="26.0" layoutY="564.0" onAction="#onbtNewAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

<JFXButton fx:id="bt\_cashier\_add" layoutX="126.0" layoutY="564.0" onAction="#onBtAddAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color:#615158 ;" text="Add" textFill="WHITE" />

<JFXButton fx:id="bt\_cashier\_update" layoutX="226.0" layoutY="564.0" onAction="#onbtUpdateAction" style="-fx-background-color: #615158;" text="Update" textFill="WHITE" />

</children>

</AnchorPane>

**Admin\_category.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminCategoryController">

<children>

<TableView fx:id="tb\_category" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="580.0" layoutY="1.0" style="-fx-font-size: 30;" text="Manage Category" />

<Label layoutX="20.0" layoutY="234.0" text="ID" />

<Label layoutX="20.0" layoutY="285.0" text="Name" />

<Label layoutX="20.0" layoutY="344.0" text="Date Created" />

<JFXTextField fx:id="tf\_id" layoutX="110.0" layoutY="221.0" promptText="10001" />

<JFXTextField fx:id="tf\_name" layoutX="110.0" layoutY="273.0" promptText="Medicine" />

<JFXTextField fx:id="tf\_date\_created" editable="false" layoutX="110.0" layoutY="332.0" promptText="00/00/000" />

<JFXButton fx:id="bt\_add" layoutX="118.0" layoutY="437.0" onAction="#onAddAction" prefHeight="27.0" prefWidth="62.0" style="-fx-background-color: #615158;" text="Add" textFill="WHITE" />

<JFXButton fx:id="bt\_update" layoutX="213.0" layoutY="437.0" onAction="#onUpdateAction" prefHeight="27.0" prefWidth="71.0" style="-fx-background-color: #615158;" text="Update" textFill="WHITE" />

<JFXButton fx:id="bt\_new" layoutX="31.0" layoutY="437.0" onAction="#onbtnewAction" prefHeight="27.0" prefWidth="62.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

</children>

</AnchorPane>

**Admin\_customer.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXRadioButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.ToggleGroup?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminCustomerController">

<children>

<TableView fx:id="tb\_customer" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="20.0" layoutY="157.0" text="ID" />

<Label layoutX="20.0" layoutY="187.0" text="Name" />

<Label layoutX="20.0" layoutY="222.0" text="Age" />

<Label layoutX="20.0" layoutY="258.0" text="Gender" />

<Label layoutX="20.0" layoutY="294.0" text="Address" />

<Label layoutX="20.0" layoutY="331.0" text="Phone" />

<Label layoutX="20.0" layoutY="368.0" text="Email" />

<Label layoutX="580.0" layoutY="1.0" style="-fx-font-size: 30;" text="Manage Customer" />

<Label layoutX="20.0" layoutY="400.0" text="Amount" />

<Label layoutX="20.0" layoutY="438.0" text="Last Date Used" />

<Label layoutX="20.0" layoutY="477.0" text="Expired Date" />

<Label layoutX="20.0" layoutY="514.0" text="Date Created" />

<Label layoutX="20.0" layoutY="552.0" text="PIN" />

<JFXTextField fx:id="tf\_id" layoutX="125.0" layoutY="148.0" prefHeight="27.0" prefWidth="201.0" promptText="11001" />

<JFXTextField fx:id="tf\_name" layoutX="125.0" layoutY="181.0" prefHeight="27.0" prefWidth="201.0" promptText="Alex" />

<JFXTextField fx:id="tf\_age" layoutX="125.0" layoutY="216.0" prefHeight="27.0" prefWidth="201.0" promptText="13" />

<JFXTextField fx:id="tf\_addr" layoutX="125.0" layoutY="281.0" prefHeight="27.0" prefWidth="201.0" promptText="Mandalay" />

<JFXTextField fx:id="tf\_phone" layoutX="125.0" layoutY="319.0" prefHeight="27.0" prefWidth="201.0" promptText="09123463033" />

<JFXTextField fx:id="tf\_mail" layoutX="125.0" layoutY="355.0" prefHeight="27.0" prefWidth="201.0" promptText="example@mail.com" />

<JFXTextField fx:id="tf\_card\_no" layoutX="10.0" layoutY="90.0" prefHeight="27.0" prefWidth="241.0" promptText="Enter Card Number" />

<JFXTextField fx:id="tf\_amount" layoutX="125.0" layoutY="393.0" prefHeight="27.0" prefWidth="201.0" promptText="50000" />

<JFXTextField fx:id="tf\_last\_date\_used" layoutX="125.0" layoutY="428.0" prefHeight="27.0" prefWidth="201.0" promptText="00/00/0000" />

<JFXTextField fx:id="tf\_pin" layoutX="120.0" layoutY="545.0" prefHeight="27.0" prefWidth="201.0" promptText="1234" />

<JFXTextField fx:id="tf\_date\_created" editable="false" layoutX="120.0" layoutY="507.0" prefHeight="27.0" prefWidth="201.0" promptText="00/00/0000" />

<JFXRadioButton fx:id="rdo\_male" layoutX="125.0" layoutY="258.0" selected="true" text="Male">

<toggleGroup>

<ToggleGroup fx:id="gender" />

</toggleGroup>

</JFXRadioButton>

<JFXRadioButton fx:id="rdo\_female" layoutX="225.0" layoutY="258.0" text="Female" toggleGroup="$gender" />

<JFXButton fx:id="bt\_add" layoutX="126.0" layoutY="600.0" onAction="#onAddAction" prefHeight="27.0" prefWidth="58.0" style="-fx-background-color: #615158;" text="Add" textFill="WHITE" />

<JFXButton fx:id="bt\_update" layoutX="226.0" layoutY="600.0" onAction="#onUpdateAction" style="-fx-background-color: #615158;" text="Update" textFill="WHITE" />

<JFXButton fx:id="bt\_new" layoutX="26.0" layoutY="600.0" onAction="#onbtNewAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

<JFXButton fx:id="bt\_search" layoutX="250.0" layoutY="90.0" onAction="#onBtSearchAction" style="-fx-background-color: #615158;" text="Search" textFill="WHITE" />

<JFXTextField fx:id="tf\_expired\_date" layoutX="120.0" layoutY="466.0" prefHeight="27.0" prefWidth="201.0" promptText="00/00/0000" />

</children>

</AnchorPane>

**Admin\_panel.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="700.0" prefWidth="1304.0" style="-fx-background-color: #878584;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.MainAdminController">

<children>

<AnchorPane layoutX="-2.0" layoutY="-8.0" prefHeight="71.0" prefWidth="1320.0" style="-fx-background-color: #fff;">

<children>

<JFXButton fx:id="bt\_report" layoutX="1103.0" layoutY="14.0" onAction="#onReportAction" prefHeight="56.0" prefWidth="100.0" style="-fx-background-color: #000;" text="Report" textFill="WHITE" />

<JFXButton fx:id="bt\_popular" layoutX="735.0" layoutY="14.0" onAction="#onPopularAction" prefHeight="56.0" prefWidth="115.0" style="-fx-background-color: #000;" text="Popular Items" textFill="WHITE" />

<JFXButton fx:id="bt\_chart" layoutX="967.0" layoutY="14.0" onAction="#onChartAction" prefHeight="56.0" prefWidth="131.0" style="-fx-background-color: #000;" text="View Charts" textFill="WHITE" />

<JFXButton fx:id="bt\_promotion" layoutX="635.0" layoutY="14.0" onAction="#onPromotionAction" prefHeight="56.0" prefWidth="94.0" style="-fx-background-color: #000;" text="Promotions" textFill="WHITE" />

<JFXButton fx:id="bt\_category" layoutX="125.0" layoutY="14.0" onAction="#onCategoryAction" prefHeight="56.0" prefWidth="130.0" style="-fx-background-color: #000;" text="Product Category" textFill="WHITE" />

<JFXButton fx:id="bt\_customer" layoutX="389.0" layoutY="14.0" onAction="#onCustomerAction" prefHeight="56.0" prefWidth="133.0" style="-fx-background-color: #000;" text="Manage Customer" textFill="WHITE" />

<JFXButton fx:id="bt\_logout" layoutX="1209.0" layoutY="14.0" onAction="#onLogoutAction" prefHeight="56.0" prefWidth="93.0" style="-fx-background-color: #610606;" text="Log Out" textFill="#ffffff" />

<JFXButton fx:id="bt\_product" layoutX="17.0" layoutY="14.0" onAction="#onProductAction" prefHeight="56.0" prefWidth="101.0" ripplerFill="#f2f2f2" style="-fx-background-color: #000;" text="Product Item" textFill="WHITE" />

<JFXButton fx:id="bt\_cashier" layoutX="260.0" layoutY="14.0" onAction="#onCashierAction" prefHeight="56.0" prefWidth="124.0" style="-fx-background-color: #000;" text="Manage Cashier" textFill="WHITE" />

<JFXButton fx:id="bt\_supplier" layoutX="855.0" layoutY="14.0" onAction="#onSupplierAction" prefHeight="56.0" prefWidth="107.0" style="-fx-background-color: #000;" text="Supplier" textFill="WHITE" />

<JFXButton fx:id="bt\_card" layoutX="528.0" layoutY="14.0" onAction="#onManageCardAction" prefHeight="56.0" prefWidth="101.0" style="-fx-background-color: #000;" text="Manage Card" textFill="WHITE" />

</children>

</AnchorPane>

<AnchorPane fx:id="common\_pane" layoutX="-3.0" layoutY="57.0" prefHeight="660.0" prefWidth="1320.0" style="-fx-background-color: #fff;" />

</children>

</AnchorPane>

**Admin\_popular\_item.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminPopularItemController">

<children>

<TableView fx:id="tb\_popular" layoutX="50.0" layoutY="50.0" prefHeight="590.0" prefWidth="1230.0" />

<Label layoutX="400.0" layoutY="1.0" style="-fx-font-size: 30;" text="Popular Items Over The Store" />

</children>

</AnchorPane>

**admin\_product.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXComboBox?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminProductController">

<children>

<TableView fx:id="tb\_product\_item" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="15.0" layoutY="222.0" text="Barcode" />

<JFXTextField fx:id="tf\_barcode" editable="true" layoutX="120.0" layoutY="206.0" prefHeight="27.0" prefWidth="166.0" promptText="777 xxx xxx" />

<Label layoutX="15.0" layoutY="263.0" text="Name" />

<JFXTextField fx:id="tf\_name" layoutX="120.0" layoutY="247.0" prefHeight="27.0" prefWidth="166.0" promptText="Water" />

<Label layoutX="15.0" layoutY="304.0" text="Category Name" />

<Label layoutX="15.0" layoutY="347.0" text="Price" />

<JFXTextField fx:id="tf\_price" layoutX="120.0" layoutY="336.0" prefHeight="27.0" prefWidth="166.0" promptText="1000" />

<Label layoutX="15.0" layoutY="388.0" text="Supplier Name" />

<Label layoutX="15.0" layoutY="432.0" text="Date Added" />

<JFXTextField fx:id="tf\_date\_added" layoutX="120.0" layoutY="416.0" prefHeight="27.0" prefWidth="166.0" promptText="dd/mm/yyyy" />

<Label layoutX="15.0" layoutY="476.0" text="Stock Amount" />

<JFXTextField fx:id="tf\_stock" layoutX="120.0" layoutY="460.0" prefHeight="27.0" prefWidth="166.0" promptText="300" />

<Label layoutX="15.0" layoutY="522.0" text="Expired Date" />

<JFXTextField fx:id="tf\_expired\_date" layoutX="120.0" layoutY="506.0" prefHeight="27.0" prefWidth="166.0" promptText="dd/mm/yyyy" />

<JFXButton fx:id="bt\_add" layoutX="122.0" layoutY="568.0" onAction="#onbtAddAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color: #615158;" text="Add" textFill="#fcfafa" />

<JFXButton fx:id="bt\_update" layoutX="221.0" layoutY="568.0" onAction="#onBtUpdateAction" style="-fx-background-color: #615158;" text="Update" textFill="WHITE" />

<Label layoutX="491.0" style="-fx-font-size: 30;" text="Manage Product Items" />

<JFXComboBox fx:id="cbo\_category" layoutX="120.0" layoutY="298.0" prefHeight="27.0" prefWidth="166.0" promptText="Snack" style="-fx-background-color: #ededed;" />

<JFXComboBox fx:id="cbo\_supplier" layoutX="120.0" layoutY="378.0" prefHeight="27.0" prefWidth="166.0" style="-fx-background-color: #ededed;" />

<JFXButton fx:id="bt\_new" layoutX="27.0" layoutY="568.0" onAction="#onbtNewAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

<JFXTextField fx:id="tf\_name\_search" layoutX="21.0" layoutY="66.0" onKeyReleased="#onNameSearchAction" prefHeight="27.0" prefWidth="266.0" promptText="Search By Name" />

<JFXTextField fx:id="tf\_barcode\_search" layoutX="22.0" layoutY="131.0" onKeyReleased="#onBarcodeSearchActionn" prefHeight="27.0" prefWidth="266.0" promptText="Search By Barcode" />

</children>

</AnchorPane>

**Admin\_promotion.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminPromotionController">

<children>

<TableView fx:id="tb\_promo" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="580.0" layoutY="1.0" style="-fx-font-size: 30;" text="Manage Promotion" />

<Label layoutX="30.0" layoutY="197.0" text="ID" />

<Label layoutX="30.0" layoutY="242.0" text="Promotion Name" />

<Label layoutX="30.0" layoutY="293.0" text="Product ID" />

<Label layoutX="140.0" layoutY="421.0" style="-fx-font-size: 18;" text="More" />

<JFXTextField fx:id="tf\_id" layoutX="145.0" layoutY="187.0" prefHeight="27.0" prefWidth="171.0" promptText="10001" />

<JFXTextField fx:id="tf\_name" layoutX="145.0" layoutY="229.0" prefHeight="27.0" prefWidth="171.0" promptText="Seasonal Promotion" />

<JFXTextField fx:id="tf\_product\_id" layoutX="145.0" layoutY="280.0" onAction="#onProductIDSearch" prefHeight="27.0" prefWidth="171.0" promptText="777 xxx xxx" />

<JFXTextField fx:id="tf\_percentage" layoutX="145.0" layoutY="377.0" prefHeight="27.0" prefWidth="171.0" promptText="15" />

<JFXTextField fx:id="tf\_buy" alignment="BOTTOM\_RIGHT" layoutX="85.0" layoutY="447.0" prefColumnCount="10" prefHeight="27.0" prefWidth="71.0" promptText="0" />

<JFXTextField fx:id="tf\_get" alignment="BOTTOM\_RIGHT" layoutX="227.0" layoutY="447.0" prefColumnCount="10" prefHeight="27.0" prefWidth="71.0" promptText="0" />

<Label layoutX="40.0" layoutY="452.0" text="Buy" />

<Label layoutX="179.0" layoutY="452.0" text="Get" />

<JFXButton fx:id="bt\_add" layoutX="140.0" layoutY="500.0" onAction="#onAddAction" prefHeight="27.0" prefWidth="58.0" style="-fx-background-color: #615158;" text="Add" textFill="#f8f8f8" />

<JFXButton layoutX="240.0" layoutY="500.0" onAction="#onUpdateAction" style="-fx-background-color: #615158;" text="Update" textFill="WHITE" />

<Label layoutX="30.0" layoutY="390.0" text="Percentage" />

<JFXTextField fx:id="tf\_product\_name" layoutX="145.0" layoutY="327.0" prefHeight="27.0" prefWidth="171.0" promptText="Water" />

<Label layoutX="30.0" layoutY="346.0" text="Product Name" />

<JFXButton fx:id="bt\_new" layoutX="40.0" layoutY="500.0" onAction="#onNewAction" prefHeight="27.0" prefWidth="58.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

</children>

</AnchorPane>

**Admin\_supplier.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminSupplierController">

<children>

<TableView fx:id="tb\_supplier" layoutX="330.0" layoutY="50.0" prefHeight="590.0" prefWidth="970.0" />

<Label layoutX="580.0" layoutY="1.0" style="-fx-font-size: 30;" text="Manage Supplier" />

<Label layoutX="15.0" layoutY="237.0" text="ID" />

<Label layoutX="15.0" layoutY="282.0" text="Company Name" />

<Label layoutX="15.0" layoutY="336.0" text="Last Supplied Date" />

<JFXButton fx:id="bt\_add" layoutX="130.0" layoutY="427.0" onAction="#onAddAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color: #615158;" text="Add" textFill="WHITE" />

<JFXButton fx:id="bt\_update" layoutX="230.0" layoutY="427.0" onAction="#onUpdateAction" style="-fx-background-color: #615158;" text="Update" textFill="WHITE" />

<JFXTextField fx:id="tf\_id" layoutX="140.0" layoutY="232.0" prefHeight="27.0" prefWidth="171.0" promptText="10001" />

<JFXTextField fx:id="tf\_name" layoutX="140.0" layoutY="277.0" prefHeight="27.0" prefWidth="171.0" promptText="Unit Company" />

<JFXTextField fx:id="tf\_supplied\_date" layoutX="140.0" layoutY="328.0" prefHeight="27.0" prefWidth="171.0" promptText="dd/mm/yyyy" />

<JFXButton fx:id="bt\_new" layoutX="30.0" layoutY="427.0" onAction="#onNewAction" prefHeight="27.0" prefWidth="63.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

</children>

</AnchorPane>

**Admin\_view\_chart.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminViewChartController">

<children>

<AnchorPane fx:id="ch\_dailySale" layoutX="1.0" layoutY="6.0" prefHeight="322.0" prefWidth="335.0" style="-fx-background-color: #e2edff;" />

<AnchorPane fx:id="ch\_month" layoutX="60.0" layoutY="330.0" prefHeight="314.0" prefWidth="350.0" style="-fx-background-color: #e2edff;" />

<AnchorPane fx:id="ch\_category" layoutX="517.0" layoutY="7.0" prefHeight="322.0" prefWidth="350.0" style="-fx-background-color: #e2edff;" />

<AnchorPane fx:id="ch\_customer\_age" layoutX="945.0" layoutY="6.0" prefHeight="331.0" prefWidth="293.0" style="-fx-background-color: #e2edff;" />

<AnchorPane fx:id="ch\_cash\_vs\_card" layoutX="724.0" layoutY="333.0" prefHeight="322.0" prefWidth="358.0" style="-fx-background-color: #e2edff;" />

</children>

</AnchorPane>

**Admin\_view\_report.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="660.0" prefWidth="1300.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.AdminViewReportController">

<children>

<Label layoutX="500.0" layoutY="30.0" style="-fx-font-size: 40;" text="Report Generator">

<font>

<Font size="40.0" />

</font>

</Label>

<JFXButton fx:id="bt\_daily\_re" layoutX="391.0" layoutY="125.0" onAction="#onDailyAction" prefHeight="87.0" prefWidth="519.0" style="-fx-background-color: #615158;" text="Generate Daily Report" textFill="WHITE">

<font>

<Font size="29.0" />

</font>

</JFXButton>

<JFXButton fx:id="bt\_month\_re" layoutX="391.0" layoutY="230.0" onAction="#onMonthlyAction" prefHeight="87.0" prefWidth="519.0" style="-fx-background-color: #615158;" text="Generate Monthly Report" textFill="WHITE">

<font>

<Font size="29.0" />

</font>

</JFXButton>

<JFXButton fx:id="bt\_popu\_re" layoutX="391.0" layoutY="336.0" onAction="#onPopularAction" prefHeight="87.0" prefWidth="519.0" style="-fx-background-color: #615158;" text="Generate Popular Item" textFill="WHITE">

<font>

<Font size="29.0" />

</font>

</JFXButton>

</children>

</AnchorPane>

**Card\_pay.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="344.0" prefWidth="531.0" style="-fx-background-color: #c1b44d;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.CardPayController">

<children>

<Label layoutX="120.0" layoutY="77.0" text="Purchase ID" />

<Label layoutX="205.0" layoutY="27.0" style="-fx-font-size: 21;" text="Card Payment" />

<JFXTextField fx:id="tf\_purchase\_id" editable="false" layoutX="237.0" layoutY="65.0" prefHeight="27.0" prefWidth="211.0" promptText="10001" />

<Label layoutX="120.0" layoutY="127.0" text="Total Amount" />

<JFXTextField fx:id="tf\_total\_amount" alignment="BOTTOM\_RIGHT" editable="false" layoutX="236.0" layoutY="114.0" prefHeight="27.0" prefWidth="211.0" promptText="5000" />

<Label layoutX="120.0" layoutY="181.0" text="Card Number" />

<JFXTextField fx:id="tf\_card\_no" editable="false" layoutX="238.0" layoutY="166.0" prefHeight="27.0" prefWidth="211.0" promptText="777 777 xxx xxx " />

<JFXButton fx:id="bt\_pay" layoutX="408.0" layoutY="275.0" onAction="#onbtPayAction" prefHeight="37.0" prefWidth="74.0" style="-fx-background-color: #615158;" text="Pay" textFill="WHITE" />

<Label layoutX="70.0" layoutY="285.0" text="Card user will get 15% discount over total amount." textFill="#5e5c5c" />

<Label layoutX="120.0" layoutY="228.0" text="Pay Amount" />

<JFXTextField fx:id="tf\_pay\_amount" editable="false" layoutX="237.0" layoutY="215.0" prefHeight="27.0" prefWidth="211.0" />

</children>

</AnchorPane>

**Card\_redeem.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXRadioButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.ToggleGroup?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="444.0" prefWidth="562.0" style="-fx-background-color: #a59f2e;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.CardRedeemController">

<children>

<JFXTextField fx:id="tf\_qr\_search" focusColor="#2d43bf" layoutX="30.0" layoutY="25.0" onAction="#onTfQRSearchAction" promptText="QR Code search" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_customer\_name" focusColor="#2d43bf" layoutX="384.0" layoutY="7.0" promptText="Cusomter Name" unFocusColor="#0d0d0d" />

<Label layoutX="15.0" layoutY="100.0" text="Phone" textFill="#000000" />

<Label layoutX="15.0" layoutY="149.0" text="Address" textFill="#000000" />

<Label layoutX="15.0" layoutY="200.0" text="Email" textFill="#000000" />

<Label layoutX="15.0" layoutY="248.0" text="PIN" textFill="#000000" />

<Label layoutX="15.0" layoutY="343.0" text="Top Up" textFill="#000000" />

<Label layoutX="15.0" layoutY="295.0" text="Amount" textFill="#000000" />

<JFXButton fx:id="bt\_apply" layoutX="450.0" layoutY="396.0" onAction="#onBtApplyAction" prefHeight="27.0" prefWidth="61.0" style="-fx-background-color: #615158;" text="Apply" textFill="#f2f2f2" />

<JFXButton fx:id="bt\_cancel" layoutX="350.0" layoutY="396.0" onAction="#onBtCancelAction" prefHeight="27.0" prefWidth="65.0" style="-fx-background-color: #615158;" text="Cancel" textFill="#f2f2f2" />

<JFXTextField fx:id="tf\_phone" focusColor="#2d43bf" layoutX="90.0" layoutY="84.0" promptText="0912344567" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_address" focusColor="#2d43bf" layoutX="90.0" layoutY="134.0" promptText="Mandalay" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_email" focusColor="#2d43bf" layoutX="90.0" layoutY="185.0" promptText="example@mail.com" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_pin" editable="false" focusColor="#2d43bf" layoutX="90.0" layoutY="233.0" promptText="1234" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_amount" editable="false" focusColor="#2d43bf" layoutX="90.0" layoutY="279.0" promptText="50000" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_top\_up" focusColor="#2d43bf" layoutX="90.0" layoutY="326.0" promptText="50000" unFocusColor="#0d0d0d" />

<JFXButton fx:id="bt\_redeem" layoutX="250.0" layoutY="396.0" onAction="#onBtRedeemAction" ripplerFill="#f7f8fc" style="-fx-background-color: #615158;" text="Top up" textFill="#fff7f7" />

<Label layoutX="300.0" layoutY="100.0" text="Gender" textFill="#000000" />

<Label layoutX="300.0" layoutY="149.0" text="Age" textFill="#000000" />

<Label layoutX="300.0" layoutY="200.0" text="Date Created" textFill="#000000" />

<Label layoutX="300.0" layoutY="248.0" text="Expired date" textFill="#000000" />

<Label layoutX="300.0" layoutY="295.0" text="Last Used" textFill="#000000" />

<JFXTextField fx:id="tf\_age" focusColor="#2d43bf" layoutX="380.0" layoutY="133.0" promptText="13" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_date\_created" editable="false" focusColor="#2d43bf" layoutX="378.0" layoutY="185.0" promptText="dd/mm/yyyy" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_expired\_date" editable="false" focusColor="#2d43bf" layoutX="377.0" layoutY="233.0" promptText="dd/mm/yyyy" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_last\_used" editable="false" focusColor="#2d43bf" layoutX="376.0" layoutY="279.0" promptText="dd/mm/yyyy" unFocusColor="#0d0d0d" />

<JFXRadioButton fx:id="rdo\_male" layoutX="376.0" layoutY="103.0" text="male" textFill="#000000">

<toggleGroup>

<ToggleGroup fx:id="gender" />

</toggleGroup>

</JFXRadioButton>

<JFXRadioButton fx:id="rdo\_female" layoutX="452.0" layoutY="103.0" text="female" textFill="#000000" toggleGroup="$gender" />

<Label layoutX="49.0" layoutY="7.0" text="Search Here" textFill="#000000" />

</children>

</AnchorPane>

**cashier\_main.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.ScrollPane?>

<?import javafx.scene.control.SplitPane?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="700.0" prefWidth="1300.0" style="-fx-background-color: #FFF; -fx-font-size: 25;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.MainCashierController">

<children>

<AnchorPane layoutY="-10.0" prefHeight="70.0" prefWidth="1300.0" style="-fx-background-color: #21000e;">

<children>

<ImageView fitHeight="48.0" fitWidth="57.0" layoutX="51.0" layoutY="15.0" pickOnBounds="true" preserveRatio="true">

<image>

<Image url="@../graphic/poslogorect.png" />

</image>

</ImageView>

<Label layoutX="141.0" layoutY="9.0" prefHeight="61.0" prefWidth="332.0" style="-fx-text-fill: #FFF; -fx-font-size: 35;" text="MxM Sale System" />

<Label fx:id="lb\_cashier\_name" layoutX="936.0" layoutY="27.0" style="-fx-text-fill: #FFF; -fx-font-size: 25;" text="Cashier Name" />

<JFXButton fx:id="bt\_logout" layoutX="1130.0" layoutY="15.0" onAction="#onLogoutClick" prefHeight="40.0" prefWidth="155.0" style="-fx-background-color: #610606;" text="Log Out" textFill="WHITE" />

</children>

</AnchorPane>

<SplitPane dividerPositions="0.5833333333333334" layoutX="4.0" layoutY="64.0" prefHeight="642.0" prefWidth="1343.0">

<items>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="600.0" prefWidth="1300.0">

<children>

<JFXTextField fx:id="tf\_barcode\_search" layoutX="18.0" layoutY="3.0" onKeyReleased="#tfTypeSearchAction" prefHeight="48.0" prefWidth="247.0" promptText="Search by barcode" unFocusColor="#252525" />

<JFXButton fx:id="bt\_create\_card" layoutX="590.0" layoutY="571.0" onAction="#onbtCreateCardClick" style="-fx-background-color: #000; -fx-text-fill: #FFF;" text="Create Card" />

<TableView fx:id="tb\_total\_item" layoutX="11.0" layoutY="58.0" prefHeight="504.0" prefWidth="758.0" />

<JFXTextField fx:id="tf\_name\_search" layoutX="312.0" layoutY="3.0" onKeyReleased="#tfNameSearchAction" promptText="Search by name" />

<JFXButton fx:id="bt\_redeem" layoutX="38.0" layoutY="571.0" onAction="#onbtRedeemClick" style="-fx-background-color: #000; -fx-text-fill: #FFF;" text="Topup Card" />

</children>

</AnchorPane>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="675.0" prefWidth="561.0">

<children>

<AnchorPane layoutX="-2.0" layoutY="-7.0" prefHeight="70.0" prefWidth="508.0" style="-fx-background-color: #21000e;">

<children>

<Label layoutX="215.0" layoutY="6.0" prefHeight="35.0" prefWidth="120.0" style="-fx-font-size: 20;" text="Retail Store" textFill="#f8f8f8" />

<Label layoutX="211.0" layoutY="29.0" prefHeight="49.0" prefWidth="127.0" text="Sale Items" textFill="#f2f2f2" />

<Label layoutX="9.0" layoutY="54.0" style="-fx-font-size: 11;" text="Slip No:" textFill="#f5f5f5" />

<Label fx:id="lb\_slip\_no" layoutX="62.0" layoutY="53.0" style="-fx-font-size: 13;" text="113" textFill="WHITE" />

<JFXButton fx:id="bt\_new" graphicTextGap="3.0" layoutX="392.0" layoutY="15.0" onAction="#onbtNewClick" prefHeight="48.0" prefWidth="104.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

</children>

</AnchorPane>

<ScrollPane layoutX="-2.0" layoutY="71.0" prefHeight="416.0" prefWidth="508.0" style="-fx-background-color: #21000e;">

<content>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="413.0" prefWidth="427.0">

<children>

<TableView fx:id="tb\_sale" prefHeight="411.0" prefWidth="508.0" />

</children>

</AnchorPane>

</content>

</ScrollPane>

<JFXButton fx:id="bt\_pay" layoutX="384.0" layoutY="500.0" onAction="#onBtPayClick" prefHeight="40.0" prefWidth="104.0" style="-fx-background-color: #057a3a; -fx-font-size: 16;" text="Card Pay" textFill="WHITE" />

<Label layoutX="20.0" layoutY="500.0" style="-fx-font-size: 18;" text="Total" />

<JFXTextField fx:id="tf\_total" editable="false" layoutX="145.0" layoutY="484.0" prefHeight="48.0" prefWidth="220.0" />

<Label layoutX="20.0" layoutY="550.0" style="-fx-font-size: 18;" text="Pay Amount" />

<JFXTextField fx:id="tf\_pay\_amount" layoutX="145.0" layoutY="529.0" onAction="#onEnterButtonClick" prefHeight="48.0" prefWidth="220.0" />

<Label layoutX="20.0" layoutY="600.0" style="-fx-font-size: 18;" text="Change" />

<JFXTextField fx:id="tf\_change" editable="false" layoutX="145.0" layoutY="572.0" prefHeight="48.0" prefWidth="220.0" />

<JFXButton fx:id="btPrint" layoutX="384.0" layoutY="567.0" onAction="#onbtPrintClick" prefHeight="40.0" prefWidth="104.0" style="-fx-background-color: #107ac2; -fx-font-size: 16;" text="Print" textFill="WHITE" />

</children></AnchorPane>

</items>

</SplitPane>

</children>

</AnchorPane>

**Create\_card.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXRadioButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.ToggleGroup?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="444.0" prefWidth="562.0" style="-fx-background-color: #a59f2e;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.CreateCardController">

<children>

<JFXTextField fx:id="tf\_phone" focusColor="#6571c2" layoutX="90.0" layoutY="134.0" promptText="0912344578" unFocusColor="#000000" />

<Label layoutX="15.0" layoutY="147.0" text="Phone" textFill="#000000" />

<Label layoutX="15.0" layoutY="193.0" text="Address" textFill="#000000" />

<Label layoutX="15.0" layoutY="236.0" text="Email" textFill="#000000" />

<Label layoutX="15.0" layoutY="284.0" text="PIN" textFill="#000000" />

<Label layoutX="15.0" layoutY="374.0" text="Top Up" textFill="#000000" />

<Label layoutX="15.0" layoutY="326.0" text="Amount" textFill="#000000" />

<JFXButton fx:id="bt\_create" layoutX="500.0" layoutY="400.0" onAction="#onBtCreateAction" prefHeight="27.0" prefWidth="70.0" style="-fx-background-color: #615158;" text="Create" textFill="#f2f2f2" />

<JFXButton fx:id="bt\_cancel" layoutX="400.0" layoutY="400.0" onAction="#onBtCancelAction" prefHeight="27.0" prefWidth="70.0" style="-fx-background-color: #615158;" text="Cancel" textFill="#f2f2f2" />

<JFXTextField fx:id="tf\_name" layoutX="90.0" layoutY="82.0" promptText="Alex" unFocusColor="#252424" />

<JFXTextField fx:id="tf\_address" layoutX="90.0" layoutY="178.0" promptText="Mandalay" unFocusColor="#1e1e1e" />

<JFXTextField fx:id="tf\_email" layoutX="90.0" layoutY="221.0" promptText="example@gmail.com" unFocusColor="#0d0d0d" />

<JFXTextField fx:id="tf\_pin" editable="false" layoutX="90.0" layoutY="269.0" promptText="1234" unFocusColor="#1b1a1a" />

<JFXTextField fx:id="tf\_amount" editable="false" layoutX="90.0" layoutY="310.0" promptText="50000" unFocusColor="#141414" />

<JFXTextField fx:id="tf\_top\_up" layoutX="90.0" layoutY="357.0" promptText="50000" unFocusColor="#141414" />

<JFXButton fx:id="bt\_redeem" layoutX="200.0" layoutY="400.0" onAction="#onBtRedeemAction" ripplerFill="#f7f8fc" style="-fx-background-color: #615158;" text="Redeem" textFill="#fff7f7" />

<Label layoutX="290.0" layoutY="95.0" text="Gender" />

<Label layoutX="290.0" layoutY="147.0" text="Age" />

<Label layoutX="290.0" layoutY="193.0" text="Date Created" />

<Label layoutX="290.0" layoutY="236.0" text="Expired date" />

<JFXTextField fx:id="tf\_age" layoutX="376.0" layoutY="125.0" promptText="13" />

<JFXRadioButton fx:id="rdo\_male" layoutX="376.0" layoutY="95.0" selected="true" text="male">

<toggleGroup>

<ToggleGroup fx:id="gender" />

</toggleGroup>

</JFXRadioButton>

<JFXRadioButton fx:id="rdo\_female" layoutX="452.0" layoutY="95.0" text="female" toggleGroup="$gender" />

<Label layoutX="15.0" layoutY="95.0" text="Name" />

<JFXTextField fx:id="tf\_date\_created" editable="false" layoutX="378.0" layoutY="170.0" promptText="dd/mm/yyyy" />

<Label layoutX="229.0" layoutY="4.0" style="-fx-font-size: 18;" text="Register Form" />

<Label layoutX="266.0" layoutY="41.0" text="Card Number" />

<JFXTextField fx:id="tf\_card\_number" layoutX="370.0" layoutY="26.0" onAction="#tfCardNumberAction" prefHeight="27.0" prefWidth="170.0" />

<Label layoutX="17.0" layoutY="42.0" text="Customer ID - " />

<Label fx:id="lb\_new\_id" layoutX="110.0" layoutY="42.0" text="1\*\*\*\*\*" />

<JFXButton fx:id="bt\_new" layoutX="300.0" layoutY="400.0" onAction="#onNewAction" prefHeight="27.0" prefWidth="58.0" style="-fx-background-color: #615158;" text="New" textFill="WHITE" />

<JFXTextField fx:id="tf\_expired\_date" layoutX="378.0" layoutY="221.0" prefHeight="27.0" prefWidth="151.0" promptText="dd/mm/yyyy" />

</children>

</AnchorPane>

**Page\_login.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXPasswordField?>

<?import com.jfoenix.controls.JFXRadioButton?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.ToggleGroup?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="700.0" prefWidth="1320.0" style="-fx-background-color: #873d3d;" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.LoginController">

<children>

<AnchorPane layoutX="810.0" layoutY="75.0" prefHeight="548.0" prefWidth="447.0" style="-fx-background-color: #540002;">

<children>

<JFXTextField id="txt\_id" fx:id="tf\_id" focusColor="#1749ad" layoutX="84.0" layoutY="215.0" prefHeight="40.0" prefWidth="280.0" promptText="Login ID" style="-fx-text-fill: #FFF;" unFocusColor="#bdaf97" />

<JFXPasswordField fx:id="tf\_pass" layoutX="84.0" layoutY="303.0" prefHeight="47.0" prefWidth="286.0" promptText="Password" style="-fx-text-fill: #FFF;" unFocusColor="#bdaf97" />

<JFXButton fx:id="bt\_login" defaultButton="true" layoutX="88.0" layoutY="443.0" onAction="#bt\_login\_action" prefHeight="49.0" prefWidth="278.0" ripplerFill="#f3f3d0" style="-fx-background-color: #e8d5c5;" text="Login" />

<ImageView fitHeight="162.0" fitWidth="190.0" layoutX="146.0" layoutY="28.0" pickOnBounds="true" preserveRatio="true" style="-fx-alignment: center;">

<image>

<Image url="@../graphic/poslogorect.png" />

</image>

</ImageView>

<JFXRadioButton fx:id="bt\_rdo\_admin" layoutX="84.0" layoutY="399.0" text="Admin" textFill="WHITE">

<toggleGroup>

<ToggleGroup fx:id="usertype" />

</toggleGroup>

</JFXRadioButton>

<JFXRadioButton fx:id="bt\_rdo\_cashier" layoutX="266.0" layoutY="399.0" text="Cashier" textFill="#f2f2f2" toggleGroup="$usertype" />

</children>

</AnchorPane>

<Label layoutX="131.0" layoutY="284.0" prefHeight="114.0" prefWidth="570.0" style="-fx-text-fill: #000; -fx-font-size: 60; -fx-font-style: italic;" text="MxM's Sale System" />

<Label layoutX="200.0" layoutY="391.0" prefHeight="59.0" prefWidth="500.0" style="-fx-text-fill: #000; -fx-font-size: 27;" text="Made For Your Retail and sale Operation" textAlignment="JUSTIFY" />

<Label layoutX="600.0" layoutY="435.0" text="ALL IN ONE" />

<Label layoutX="980.0" layoutY="669.0" text="Developed by Mokhtari and Merzougui Â©" />

<Label layoutX="19.0" layoutY="669.0" prefHeight="15.0" prefWidth="500.0" text=" Copyright Â© to MxM2021" />

</children>

</AnchorPane>

**Forecasting**

import warnings

import itertools

import numpy as np

import matplotlib.pyplot as plt

warnings.filterwarnings("ignore")

plt.style.use('fivethirtyeight')

import pandas as pd

import statsmodels.api as sm

import matplotlib

matplotlib.rcParams['axes.labelsize'] = 14

matplotlib.rcParams['xtick.labelsize'] = 12

matplotlib.rcParams['ytick.labelsize'] = 12

matplotlib.rcParams['text.color'] = 'k'

df = pd.read\_excel("Superstore.xls")

tech = df.loc[df['Category'] == 'Technology']

tech['Order Date'].min(), tech['Order Date'].max()

cols = ['Row ID', 'Order ID', 'Ship Date old', 'Ship Mode', 'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Quantity', 'Discount', 'Profit']

tech.drop(cols, axis=1, inplace=True)

tech = tech.sort\_values('Order Date')

tech.isnull().sum()

tech = tech.groupby('Order Date')['Sales'].sum().reset\_index()

tech = tech.set\_index('Order Date')

tech.index

y = tech['Sales'].resample('MS').mean()

y['2017':]

y['2016':]

y.plot(figsize=(15, 6))

plt.show()

from pylab import rcParams

rcParams['figure.figsize'] = 18, 8

decomposition = sm.tsa.seasonal\_decompose(y, model='additive')

fig = decomposition.plot()

plt.show()

p = d = q = range(0, 2)

pdq = list(itertools.product(p, d, q))

seasonal\_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')

print('SARIMAX: {} x {}'.format(pdq[1], seasonal\_pdq[1]))

print('SARIMAX: {} x {}'.format(pdq[1], seasonal\_pdq[2]))

print('SARIMAX: {} x {}'.format(pdq[2], seasonal\_pdq[3]))

print('SARIMAX: {} x {}'.format(pdq[2], seasonal\_pdq[4]))

mod = sm.tsa.statespace.SARIMAX(y,

order=(1, 1, 1),

seasonal\_order=(1, 1, 0, 12),

enforce\_stationarity=False,

enforce\_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])

results.plot\_diagnostics(figsize=(16, 8))

plt.show()

pred = results.get\_prediction(start=pd.to\_datetime('2022-01-01'), dynamic=False)

pred\_ci = pred.conf\_int()

ax = y['2020':].plot(label='observed')

pred.predicted\_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))

ax.fill\_between(pred\_ci.index,

pred\_ci.iloc[:, 0],

pred\_ci.iloc[:, 1], color='k', alpha=.2)

ax.set\_xlabel('Date')

ax.set\_ylabel('Furniture Sales')

plt.legend()

plt.show()

y\_forecasted = pred.predicted\_mean

y\_truth = y['2017-01-01':]

mse = ((y\_forecasted - y\_truth) \*\* 2).mean()

print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))

print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))

pred\_uc = results.get\_forecast(steps=100)

pred\_ci = pred\_uc.conf\_int()

ax = y.plot(label='observed', figsize=(14, 7))

pred\_uc.predicted\_mean.plot(ax=ax, label='Forecast')

ax.fill\_between(pred\_ci.index,

pred\_ci.iloc[:, 0],

pred\_ci.iloc[:, 1], color='k', alpha=.25)

ax.set\_xlabel('Date')

ax.set\_ylabel('Technology Sales')

plt.legend()

plt.show()

tech = df.loc[df['Category'] == 'Technology']

office = df.loc[df['Category'] == 'Office Supplies']

tech.shape, office.shape

cols = ['Row ID', 'Order ID', 'Ship Date old', 'Ship Mode', 'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Quantity', 'Discount', 'Profit']

tech.drop(cols, axis=1, inplace=True)

office.drop(cols, axis=1, inplace=True)

tech = tech.sort\_values('Order Date')

office = office.sort\_values('Order Date')

tech = tech.groupby('Order Date')['Sales'].sum().reset\_index()

office = office.groupby('Order Date')['Sales'].sum().reset\_index()

tech = tech.set\_index('Order Date')

office = office.set\_index('Order Date')

y\_tech = tech['Sales'].resample('MS').mean()

y\_office = office['Sales'].resample('MS').mean()

tech = pd.DataFrame({'Order Date':y\_tech.index, 'Sales':y\_tech.values})

office = pd.DataFrame({'Order Date': y\_office.index, 'Sales': y\_office.values})

store = tech.merge(office, how='inner', on='Order Date')

store.rename(columns={'Sales\_x': 'tech\_sales', 'Sales\_y': 'office\_sales'}, inplace=True)

store.head()