**Bus Data:**

package busticketbookingmanagementsystem;

import java.sql.Date;

public class busData {

private Integer busId;

private String location;

private String status;

private Double price;

private Date date;

public busData(Integer busId, String location, String status, Double price, Date date){

this.busId = busId;

this.location = location;

this.status = status;

this.price = price;

this.date = date;

}

public Integer getBusId(){

return busId;

}

public String getLocation(){

return location;

}

public String getStatus(){

return status;

}

public Double getPrice(){

return price;

}

public Date getDate(){

return date;

    }

}

**Bus Ticket Booking Management System:**

package busticketbookingmanagementsystem;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

public class BusTicketBookingManagementSystem extends Application {

private double x = 0;

private double y = 0;

@Override

public void start(Stage stage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

Scene scene = new Scene(root);

root.setOnMousePressed((MouseEvent event) -> {

x = event.getSceneX();

y = event.getSceneY();

});

root.setOnMouseDragged((MouseEvent event) -> {

stage.setX(event.getScreenX() - x);

stage.setY(event.getScreenY() - y);

stage.setOpacity(.8);

});

root.setOnMouseReleased((MouseEvent event) -> {

stage.setOpacity(1);

});

stage.initStyle(StageStyle.TRANSPARENT);

stage.setScene(scene);

stage.show();

}

public static void main(String[] args) {

launch(args);

    }

}

**Customer Data:**

package busticketbookingmanagementsystem;

import java.sql.Date;

public class customerData {

private Integer customerNum;

private String firstName;

private String lastName;

private String gender;

private String phoneNum;

private Integer busId;

private String location;

private String type;

private Integer seatNum;

private Double total;

private Date date;

public customerData(Integer customerNum, String firstName, String lastName, String gender, String phoneNum, Integer busId, String location, String type, Integer seatNum, Double total, Date date)

{

this.customerNum = customerNum;

this.firstName = firstName;

this.lastName = lastName;

this.gender = gender;

this.phoneNum = phoneNum;

this.busId = busId;

this.location = location;

this.type = type;

this.seatNum = seatNum;

this.total = total;

this.date = date;

}

public Integer getCustomerNum(){

return customerNum;

}

public String getFirstName(){

return firstName;

}

public String getLastName(){

return lastName;

}

public String getGender(){

return gender;

}

public String getPhoneNum(){

return phoneNum;

}

public Integer getBusId(){

return busId;

}

public String getLocation(){

return location;

}

public String getType(){

return type;

}

public Integer getSeatNum(){

return seatNum;

}

public Double getTotal(){

return total;

}

public Date getDate(){

return date;

    }

}

**Dashboard Controller:**

package busticketbookingmanagementsystem;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Optional;

import java.util.ResourceBundle;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.collections.transformation.FilteredList;

import javafx.collections.transformation.SortedList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.chart.AreaChart;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.Button;

import javafx.scene.control.ButtonType;

import javafx.scene.control.ComboBox;

import javafx.scene.control.DatePicker;

import javafx.scene.control.Label;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.AnchorPane;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import net.sf.jasperreports.engine.JasperCompileManager;

import net.sf.jasperreports.engine.JasperFillManager;

import net.sf.jasperreports.engine.JasperPrint;

import net.sf.jasperreports.engine.JasperReport;

import net.sf.jasperreports.engine.design.JasperDesign;

import net.sf.jasperreports.engine.xml.JRXmlLoader;

import net.sf.jasperreports.view.JasperViewer;

public class dashboardController implements Initializable {

@FXML

private AnchorPane main\_form;

@FXML

private Button close;

@FXML

private Button minimize;

@FXML

private Label username;

@FXML

private Button dashboard\_Btn;

@FXML

private Button availableB\_Btn;

@FXML

private Button bookingTicket\_Btn;

@FXML

private Button customers\_btn;

@FXML

private Button logout;

@FXML

private AnchorPane dashboard\_form;

@FXML

private Label dashboard\_availableB;

@FXML

private Label dashboard\_incomeToday;

@FXML

private Label dashboard\_totalIncome;

@FXML

private AreaChart<?, ?> dashboard\_chart;

@FXML

private AnchorPane availableB\_form;

@FXML

private TextField availableB\_busID;

@FXML

private TextField availableB\_location;

@FXML

private ComboBox<?> availableB\_status;

@FXML

private TextField availableB\_price;

@FXML

private DatePicker availableB\_date;

@FXML

private Button availableB\_addBtn;

@FXML

private Button availableB\_updateBtn;

@FXML

private Button availableB\_resetBtn;

@FXML

private Button availableB\_deleteBtn;

@FXML

private TableView<busData> availableB\_tableView;

@FXML

private TableColumn<busData, String> availableB\_col\_busID;

@FXML

private TableColumn<busData, String> availableB\_col\_location;

@FXML

private TableColumn<busData, String> availableB\_col\_status;

@FXML

private TableColumn<busData, String> availableB\_col\_price;

@FXML

private TableColumn<busData, String> availableB\_col\_date;

@FXML

private TextField availableB\_search;

@FXML

private AnchorPane bookingTicket\_form;

@FXML

private ComboBox<?> bookingTicket\_busId;

@FXML

private ComboBox<?> bookingTicket\_location;

@FXML

private ComboBox<?> bookingTicket\_type;

@FXML

private ComboBox<?> bookingTicket\_ticketNum;

@FXML

private TextField bookingTicket\_firstName;

@FXML

private TextField bookingTicket\_lastName;

@FXML

private ComboBox<?> bookingTicket\_gender;

@FXML

private TextField bookingTicket\_phoneNum;

@FXML

private DatePicker bookingTicket\_date;

@FXML

private Button bookingTicket\_selectBtn;

@FXML

private Button bookingTicket\_resetBtn;

@FXML

private Label bookingTicket\_sci\_firstName;

@FXML

private Label bookingTicket\_sci\_lastNmae;

@FXML

private Label bookingTicket\_sci\_gender;

@FXML

private Label bookingTicket\_sci\_phoneNum;

@FXML

private Label bookingTicket\_sci\_date;

@FXML

private Label bookingTicket\_sci\_busID;

@FXML

private Label bookingTicket\_sci\_location;

@FXML

private Label bookingTicket\_sci\_type;

@FXML

private Label bookingTicket\_sci\_ticketNum;

@FXML

private Label bookingTicket\_sci\_total;

@FXML

private Button bookingTicket\_sci\_pay;

@FXML

private Button bookingTicket\_sci\_receipt;

@FXML

private AnchorPane customer\_Form;

@FXML

private TableView<customerData> customers\_tableView;

@FXML

private TableColumn<customerData, String> customers\_customerNum;

@FXML

private TableColumn<customerData, String> customers\_ticketNum;

@FXML

private TableColumn<customerData, String> customers\_firstName;

@FXML

private TableColumn<customerData, String> customers\_lastName;

@FXML

private TableColumn<customerData, String> customers\_gender;

@FXML

private TableColumn<customerData, String> customers\_phoneNum;

@FXML

private TableColumn<customerData, String> customers\_busID;

@FXML

private TableColumn<customerData, String> customers\_location;

@FXML

private TableColumn<customerData, String> customers\_type;

@FXML

private TableColumn<customerData, String> customers\_date;

@FXML

private TextField customers\_search;

// DATABASE TOOLS

private Connection connect;

private PreparedStatement prepare;

private ResultSet result;

private Statement statement;

// LET'S WORK FOR AVAILABLE BUSES FORM FIRST :

public void availableBusAdd() {

String addData = "INSERT INTO bus (bus\_id,location,status,price,date) VALUES(?,?,?,?,?)";

connect = database.connectDb();

try {

Alert alert;

// CHECK IF THE FIELDS ARE EMPTY

if (availableB\_busID.getText().isEmpty()

|| availableB\_location.getText().isEmpty()

|| availableB\_status.getSelectionModel().getSelectedItem() == null

|| availableB\_price.getText().isEmpty()

|| availableB\_date.getValue() == null) {

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Please fill all blank fields");

alert.showAndWait();

}

else {

// CHECK IF THE BUS ID IS ALREADY EXIST

String check = "SELECT bus\_id FROM bus WHERE bus\_id = '"

+ availableB\_busID.getText() + "'";

statement = connect.createStatement();

result = statement.executeQuery(check);

if (result.next()) {

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Bus ID: " + availableB\_busID.getText() + " was already exist!");

alert.showAndWait();

} else {

prepare = connect.prepareStatement(addData);

prepare.setString(1, availableB\_busID.getText());

prepare.setString(2, availableB\_location.getText());

prepare.setString(3, (String) availableB\_status.getSelectionModel().getSelectedItem());

prepare.setString(4, availableB\_price.getText());

prepare.setString(5, String.valueOf(availableB\_date.getValue()));

prepare.executeUpdate();

alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Information Message");

alert.setHeaderText(null);

alert.setContentText("Successfully Added!");

alert.showAndWait();

// TO UPDATE YOUR TABLE VIEW ONCE THE DATA IS SUCCESSFUL

availableBShowBusData();

availableBusReset();

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void availableBusUpdate() {

String updateData = "UPDATE bus SET location = '"

+ availableB\_location.getText() + "', status = '"

+ availableB\_status.getSelectionModel().getSelectedItem()

+ "', price = '" + availableB\_price.getText()

+ "', date = '" + availableB\_date.getValue()

+ "' WHERE bus\_id = '" + availableB\_busID.getText() + "'";

connect = database.connectDb();

Alert alert;

try {

if (availableB\_busID.getText().isEmpty()

|| availableB\_location.getText().isEmpty()

|| availableB\_status.getSelectionModel().getSelectedItem() == null

|| availableB\_price.getText().isEmpty()

|| availableB\_date.getValue() == null) {

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Please select the item first");

alert.showAndWait();

} else {

alert = new Alert(AlertType.CONFIRMATION);

alert.setTitle("Confirmation Message");

alert.setHeaderText(null);

alert.setContentText("Are you sure you want to UPDATE Bus ID: " + availableB\_busID.getText() + "?");

Optional<ButtonType> option = alert.showAndWait();

if (option.get().equals(ButtonType.OK)) {

prepare = connect.prepareStatement(updateData);

prepare.executeUpdate();

alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Information Message");

alert.setHeaderText(null);

alert.setContentText("Successfully Updated!");

alert.showAndWait();

availableBShowBusData();

availableBusReset();

} else {

return;

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void availableBusDelete(){

String deleteData = "DELETE FROM bus WHERE bus\_id = '"

+availableB\_busID.getText()+"'";

connect = database.connectDb();

try{

Alert alert;

if (availableB\_busID.getText().isEmpty()

|| availableB\_location.getText().isEmpty()

|| availableB\_status.getSelectionModel().getSelectedItem() == null

|| availableB\_price.getText().isEmpty() || availableB\_date.getValue() == null)

{

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Please select the item first");

alert.showAndWait();

} else {

alert = new Alert(AlertType.CONFIRMATION);

alert.setTitle("Confirmation Message");

alert.setHeaderText(null);

alert.setContentText("Are you sure you want to delete Bus ID: " + availableB\_busID.getText() + "?");

Optional<ButtonType> option = alert.showAndWait();

if(option.get().equals(ButtonType.OK)){

statement = connect.createStatement();

statement.executeUpdate(deleteData);

alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Information Message");

alert.setHeaderText(null);

alert.setContentText("Successfully Deleted!");

alert.showAndWait();

availableBShowBusData();

availableBusReset();

}

else{

return;

}

}

// PROCEED TO BOOKING TICKET :

}catch(Exception e){

e.printStackTrace();

}

}

public void availableBusReset() {

availableB\_busID.setText("");

availableB\_location.setText("");

availableB\_status.getSelectionModel().clearSelection();

availableB\_price.setText("");

availableB\_date.setValue(null);

}

private String[] statusList = {"Available", "Not Available"};

public void comboBoxStatus() {

List<String> listS = new ArrayList<>();

for (String data : statusList) {

listS.add(data);

}

ObservableList listStatus = FXCollections.observableArrayList(listS);

availableB\_status.setItems(listStatus);

}

public ObservableList<busData> availableBusBusData() {

ObservableList<busData> busListData = FXCollections.observableArrayList();

String sql = "SELECT \* FROM bus";

connect = database.connectDb();

try {

prepare = connect.prepareStatement(sql);

result = prepare.executeQuery();

busData busD;

while (result.next()) {

busD = new busData(result.getInt("bus\_id"),

result.getString("location"),

result.getString("status"),

result.getDouble("price"),

result.getDate("date"));

busListData.add(busD);

}

} catch (Exception e) {

e.printStackTrace();

}

return busListData;

}

private ObservableList<busData> availableBBusListData;

public void availableBShowBusData() {

availableBBusListData = availableBusBusData();

availableB\_col\_busID.setCellValueFactory(new PropertyValueFactory<>("busId"));

availableB\_col\_location.setCellValueFactory(new PropertyValueFactory<>("location"));

availableB\_col\_status.setCellValueFactory(new PropertyValueFactory<>("status"));

availableB\_col\_price.setCellValueFactory(new PropertyValueFactory<>("price"));

availableB\_col\_date.setCellValueFactory(new PropertyValueFactory<>("date"));

availableB\_tableView.setItems(availableBBusListData);

}

public void avaialbleBSelectBusData() {

busData busD = availableB\_tableView.getSelectionModel().getSelectedItem();

int num = availableB\_tableView.getSelectionModel().getSelectedIndex();

if ((num - 1) < -1) {

return;

}

availableB\_busID.setText(String.valueOf(busD.getBusId()));

availableB\_location.setText(busD.getLocation());

availableB\_price.setText(String.valueOf(busD.getPrice()));

availableB\_date.setValue(LocalDate.parse(String.valueOf(busD.getDate())));

}

public void availableSearch(){

FilteredList<busData> filter = new FilteredList<>(availableBBusListData, e-> true);

availableB\_search.textProperty().addListener((Observable, oldValue, newValue) ->{

filter.setPredicate(predicateBusData ->{

if(newValue.isEmpty() || newValue == null){

return true;

}

String searchKey = newValue.toLowerCase();

// NOTHING? THEN WE NEED TO DO THIS FIRST

if(predicateBusData.getBusId().toString().contains(searchKey)){

// NOTE, IF INTEGER OR IF THE DATA TYPE IS NOT STRING, YOU MUST BE DO toString()

return true;

}

else if(predicateBusData.getLocation().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateBusData.getStatus().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateBusData.getDate().toString().contains(searchKey)){

return true;

}

else if(predicateBusData.getPrice().toString().contains(searchKey)){

return true;

}

else return false;

});

});

SortedList<busData> sortList = new SortedList<>(filter);

sortList.comparatorProperty().bind(availableB\_tableView.comparatorProperty());

availableB\_tableView.setItems(sortList);

}

public void busIdList(){

String busD = "SELECT \* FROM bus WHERE status = 'Available'";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(busD);

result = prepare.executeQuery();

ObservableList listB = FXCollections.observableArrayList();

while(result.next()){

listB.add(result.getString("bus\_id"));

}

bookingTicket\_busId.setItems(listB);

ticketNumList();

}catch(Exception e){

e.printStackTrace();

}

}

public void LocationList(){

String locationL = "SELECT \* FROM bus WHERE status = 'Available'";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(locationL);

result = prepare.executeQuery();

ObservableList listL = FXCollections.observableArrayList();

while(result.next()){

listL.add(result.getString("location"));

}

bookingTicket\_location.setItems(listL);

}catch(Exception e){e.printStackTrace();}

}

private String[] listT = {"First Class", "Economy Class"};

public void typeList(){

List<String> tList = new ArrayList<>();

for(String data : listT){

tList.add(data);

}

ObservableList listType = FXCollections.observableArrayList(tList);

bookingTicket\_type.setItems(listType);

}

public void ticketNumList(){

List<String> listTicket = new ArrayList<>();

for(int q = 1; q <= 40; q++){

listTicket.add(String.valueOf(q));

}

// 40 ARE OUR CAPACITY SEATS

String removeSeat = "SELECT seatNum FROM customer WHERE bus\_id='"

+bookingTicket\_busId.getSelectionModel().getSelectedItem()+"'";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(removeSeat);

result = prepare.executeQuery();

while(result.next()){

listTicket.remove(result.getString("seatNum"));

}

ObservableList listTi = FXCollections.observableArrayList(listTicket);

bookingTicket\_ticketNum.setItems(listTi);

}

catch(Exception e){e.printStackTrace();}

}

// NOW LET'S CREATE TABLE FOR customer:

private double priceData = 0;

private double totalP = 0;

public void bookingTicketSelect(){

String firstName = bookingTicket\_firstName.getText();

String lastName = bookingTicket\_lastName.getText();

String gender = (String)bookingTicket\_gender.getSelectionModel().getSelectedItem();

String phoneNumber = bookingTicket\_phoneNum.getText();

String date = String.valueOf(bookingTicket\_date.getValue());

String busId = (String)bookingTicket\_busId.getSelectionModel().getSelectedItem();

String location = (String)bookingTicket\_location.getSelectionModel().getSelectedItem();

String type = (String)bookingTicket\_type.getSelectionModel().getSelectedItem();

String ticketNum = (String)bookingTicket\_ticketNum.getSelectionModel().getSelectedItem();

Alert alert;

if(firstName == null || lastName == null

|| gender == null || phoneNumber == null || date == null

|| busId == null || location == null

|| type == null || ticketNum == null){

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Please fill all blank fields");

alert.showAndWait();

}

else{

String totalPrice = "SELECT price FROM bus WHERE location = '"

+location+"'";

try{

connect = database.connectDb();

prepare = connect.prepareStatement(totalPrice);

result = prepare.executeQuery();

if(result.next()){

priceData = result.getDouble("price");

}

if(type == "First Class"){

totalP = (priceData + 100);

}

else if(type == "Economy Class"){

totalP = priceData;

}

}

catch(Exception e){

e.printStackTrace();

}

bookingTicket\_sci\_total.setText("$"+String.valueOf(totalP));

bookingTicket\_sci\_firstName.setText(firstName);

bookingTicket\_sci\_lastNmae.setText(lastName);

bookingTicket\_sci\_gender.setText(gender);

bookingTicket\_sci\_phoneNum.setText(phoneNumber);

bookingTicket\_sci\_date.setText(date);

bookingTicket\_sci\_busID.setText(busId);

bookingTicket\_sci\_location.setText(location);

bookingTicket\_sci\_type.setText(type);

bookingTicket\_sci\_ticketNum.setText(ticketNum);

alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Information Message");

alert.setHeaderText(null);

alert.setContentText("Successfully Selected!");

alert.showAndWait();

bookingTicketReset();

}

}

public void bookingTicketReset(){

bookingTicket\_firstName.setText("");

bookingTicket\_lastName.setText("");

bookingTicket\_gender.getSelectionModel().clearSelection();

bookingTicket\_phoneNum.setText("");

bookingTicket\_date.setValue(null);

}

private String[] genderL = {"Male","Female","Others"};

public void genderList(){

List<String> listG = new ArrayList<>();

for(String data : genderL){

listG.add(data);

}

ObservableList gList = FXCollections.observableArrayList(listG);

bookingTicket\_gender.setItems(gList);

}

private int countRow;

public void bookingTicketPay(){

String firstName = bookingTicket\_sci\_firstName.getText();

String lastName = bookingTicket\_sci\_lastNmae.getText();

String gender = bookingTicket\_sci\_gender.getText();

String phoneNumber = bookingTicket\_sci\_phoneNum.getText();

String date = bookingTicket\_sci\_date.getText();

String busId = bookingTicket\_sci\_busID.getText();

String location = bookingTicket\_sci\_location.getText();

String type = bookingTicket\_sci\_type.getText();

String seatNum = bookingTicket\_sci\_ticketNum.getText();

String payData = "INSERT INTO customer (customer\_id,firstName,lastName,gender,phoneNumber,bus\_id,location,type,seatNum,total,date)"

+ " VALUES(?,?,?,?,?,?,?,?,?,?,?)";

connect = database.connectDb();

try{

Alert alert;

String countNum = "SELECT COUNT(id) FROM customer";

statement = connect.createStatement();

result = statement.executeQuery(countNum);

while(result.next()){

countRow = result.getInt("COUNT(id)");

}

// CHECK IF EMPTY

if(bookingTicket\_sci\_firstName.getText().isEmpty()

|| bookingTicket\_sci\_lastNmae.getText().isEmpty()

|| bookingTicket\_sci\_gender.getText().isEmpty()

|| bookingTicket\_sci\_phoneNum.getText().isEmpty()

|| bookingTicket\_sci\_date.getText().isEmpty()

|| bookingTicket\_sci\_busID.getText().isEmpty()

|| bookingTicket\_sci\_location.getText().isEmpty()

|| bookingTicket\_sci\_type.getText().isEmpty()

|| bookingTicket\_sci\_ticketNum.getText().isEmpty()

|| totalP == 0){

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Please select the information first");

alert.showAndWait();

}

else{

alert = new Alert(AlertType.CONFIRMATION);

alert.setTitle("Confirmation Message");

alert.setHeaderText(null);

alert.setContentText("Are you sure?");

alert.showAndWait();

// NEED TO REMOVE THE SEAT# IF THE CUSTOMER IS ALREADY CHOSE THAT

prepare = connect.prepareStatement(payData);

prepare.setString(1, String.valueOf(countRow+1));

prepare.setString(2, firstName);

prepare.setString(3, lastName);

prepare.setString(4, gender);

prepare.setString(5, phoneNumber);

prepare.setString(6, busId);

prepare.setString(7, location);

prepare.setString(8, type);

prepare.setString(9, seatNum);

prepare.setString(10, String.valueOf(totalP));

prepare.setString(11, date);

prepare.executeUpdate();

String receiptData = "INSERT INTO customer\_receipt (customer\_id,total,date) VALUES(?,?,?)";

getData.number = (countRow + 1);

prepare = connect.prepareStatement(receiptData);

prepare.setString(1, String.valueOf(countRow+1));

prepare.setString(2, String.valueOf(totalP));

prepare.setString(3, date);

prepare.executeUpdate();

alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Information Message");

alert.setHeaderText(null);

alert.setContentText("Successful!");

alert.showAndWait();

// NOW LETS PROCEED TO RECEIPT

bookingTicket\_sci\_firstName.setText("");

bookingTicket\_sci\_lastNmae.setText("");

bookingTicket\_sci\_gender.setText("");

bookingTicket\_sci\_phoneNum.setText("");

bookingTicket\_sci\_date.setText("");

bookingTicket\_sci\_busID.setText("");

bookingTicket\_sci\_location.setText("");

bookingTicket\_sci\_type.setText("");

bookingTicket\_sci\_ticketNum.setText("");

bookingTicket\_sci\_total.setText("$0.0");

// WE NEED TO INSERT THE DATA ON CUSTOMER\_RECEIPT FOR OUR RECEIPT :

}

}catch(Exception e){

e.printStackTrace();

}

}

public void bookingTicketReceipt(){

HashMap hash = new HashMap();

hash.put("busD", getData.number);

try{

if(totalP > 0){

JasperDesign jDesign = JRXmlLoader.load("D:\\OneDrive\\Desktop\\Semester Notes\\Second Semester\\Java CLass Lectures\\Project\\BusManagementSystem-main\\busTicketBookingManagementSystem\\src\\busticketbookingmanagementsystem\\report.jrxml");

JasperReport jReport = JasperCompileManager.compileReport(jDesign);

JasperPrint jPrint = JasperFillManager.fillReport(jReport, hash, connect);

JasperViewer.viewReport(jPrint, false);

}else{

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Invalid :3");

alert.showAndWait();

}

}catch(Exception e){

e.printStackTrace();

}

}

// PROCEED TO CUSTOMER FORM :

public ObservableList<customerData> customersDataList(){

ObservableList<customerData> customerList = FXCollections.observableArrayList();

String sql = "SELECT \* FROM customer";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(sql);

result = prepare.executeQuery();

customerData custD;

while(result.next()){

custD = new customerData(result.getInt("customer\_id")

, result.getString("firstName")

, result.getString("lastName")

, result.getString("gender")

, result.getString("phoneNumber")

, result.getInt("bus\_id")

, result.getString("location")

, result.getString("type")

, result.getInt("seatNum")

, result.getDouble("total")

, result.getDate("date"));

customerList.add(custD);

}

}catch(Exception e){

e.printStackTrace();

}

return customerList;

}

private ObservableList<customerData> customersDataL;

public void customersShowDataList(){

customersDataL = customersDataList();

customers\_customerNum.setCellValueFactory(new PropertyValueFactory<>("customerNum"));

customers\_ticketNum.setCellValueFactory(new PropertyValueFactory<>("seatNum"));

customers\_firstName.setCellValueFactory(new PropertyValueFactory<>("firstName"));

customers\_lastName.setCellValueFactory(new PropertyValueFactory<>("lastName"));

customers\_phoneNum.setCellValueFactory(new PropertyValueFactory<>("phoneNum"));

customers\_gender.setCellValueFactory(new PropertyValueFactory<>("gender"));

customers\_busID.setCellValueFactory(new PropertyValueFactory<>("busId"));

customers\_location.setCellValueFactory(new PropertyValueFactory<>("location"));

customers\_type.setCellValueFactory(new PropertyValueFactory<>("type"));

customers\_date.setCellValueFactory(new PropertyValueFactory<>("date"));

customers\_tableView.setItems(customersDataL);

}

public void customersSearch(){

FilteredList<customerData> filter = new FilteredList<>(customersDataL, e-> true);

customers\_search.textProperty().addListener((Observable, oldValue, newValue) ->{

filter.setPredicate(predicateCustomerData ->{

if(newValue == null || newValue.isEmpty()){

return true;

}

String searchKey = newValue.toLowerCase();

if(predicateCustomerData.getCustomerNum().toString().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getSeatNum().toString().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getFirstName().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getLastName().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getGender().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getPhoneNum().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getBusId().toString().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getLocation().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getTotal().toString().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getType().toLowerCase().contains(searchKey)){

return true;

}

else if(predicateCustomerData.getDate().toString().contains(searchKey)){

return true;

}

else return false;

});

});

SortedList<customerData> sortList = new SortedList<>(filter);

sortList.comparatorProperty().bind(customers\_tableView.comparatorProperty());

customers\_tableView.setItems(sortList);

}

private double x = 0;

private double y = 0;

public void logout() {

try {

Alert alert = new Alert(AlertType.CONFIRMATION);

alert.setTitle("Confirmation Message");

alert.setHeaderText(null);

alert.setContentText("Are you sure you want to logout?");

Optional<ButtonType> option = alert.showAndWait();

if (option.get().equals(ButtonType.OK)) {

logout.getScene().getWindow().hide();

// LOGIN FORM

Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

Stage stage = new Stage();

Scene scene = new Scene(root);

root.setOnMousePressed((MouseEvent event) -> {

x = event.getSceneX();

y = event.getSceneY();

});

root.setOnMouseDragged((MouseEvent event) -> {

stage.setX(event.getScreenX() - x);

stage.setY(event.getScreenY() - y);

stage.setOpacity(.8);

});

root.setOnMouseReleased((MouseEvent event) -> {

stage.setOpacity(1);

});

stage.initStyle(StageStyle.TRANSPARENT);

stage.setScene(scene);

stage.show();

} else {

return;

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void defaultBtn() {

dashboard\_Btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #a73f4a, #3ea763)");

availableB\_Btn.setStyle("-fx-background-color:transparent");

bookingTicket\_Btn.setStyle("-fx-background-color:transparent");

customers\_btn.setStyle("-fx-background-color:transparent");

}

public void switchForm(ActionEvent event) {

if (event.getSource() == dashboard\_Btn) {

dashboard\_form.setVisible(true);

availableB\_form.setVisible(false);

bookingTicket\_form.setVisible(false);

customer\_Form.setVisible(false);

dashboard\_Btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #a73f4a, #3ea763)");

availableB\_Btn.setStyle("-fx-background-color:transparent");

bookingTicket\_Btn.setStyle("-fx-background-color:transparent");

customers\_btn.setStyle("-fx-background-color:transparent");

dashboardDisplayAB();

dashboardDisplayIT();

dashboardDisplayTI();

dashboardChart();

}

else if (event.getSource() == availableB\_Btn) {

dashboard\_form.setVisible(false);

availableB\_form.setVisible(true);

bookingTicket\_form.setVisible(false);

customer\_Form.setVisible(false);

availableB\_Btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #a73f4a, #3ea763)");

dashboard\_Btn.setStyle("-fx-background-color:transparent");

bookingTicket\_Btn.setStyle("-fx-background-color:transparent");

customers\_btn.setStyle("-fx-background-color:transparent");

// TO UPDATE THE FORM ONCE YOU CLICK THE AVAIALABLE BUSES BUTTON

availableBShowBusData();

availableSearch();

}

else if (event.getSource() == bookingTicket\_Btn) {

dashboard\_form.setVisible(false);

availableB\_form.setVisible(false);

bookingTicket\_form.setVisible(true);

customer\_Form.setVisible(false);

bookingTicket\_Btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #a73f4a, #3ea763)");

availableB\_Btn.setStyle("-fx-background-color:transparent");

dashboard\_Btn.setStyle("-fx-background-color:transparent");

customers\_btn.setStyle("-fx-background-color:transparent");

busIdList();

LocationList();

typeList();

ticketNumList();

genderList();

}

else if (event.getSource() == customers\_btn) {

dashboard\_form.setVisible(false);

availableB\_form.setVisible(false);

bookingTicket\_form.setVisible(false);

customer\_Form.setVisible(true);

customers\_btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #a73f4a, #3ea763)");

availableB\_Btn.setStyle("-fx-background-color:transparent");

bookingTicket\_Btn.setStyle("-fx-background-color:transparent");

dashboard\_Btn.setStyle("-fx-background-color:transparent");

customersShowDataList();

customersSearch();

// LET'S PROCEED TO OUR DASHBOARD FORM :

}

}

private int countAB = 0;

public void dashboardDisplayAB(){

String sql = "SELECT COUNT(id) FROM bus WHERE status = 'Available'";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(sql);

result = prepare.executeQuery();

while(result.next()){

countAB = result.getInt("COUNT(id)");

}

dashboard\_availableB.setText(String.valueOf(countAB));

}

catch(Exception e){

e.printStackTrace();

}

}

private double incomeToday = 0;

public void dashboardDisplayIT(){

Date date = new Date();

java.sql.Date sqlDate = new java.sql.Date(date.getTime());

String sql = "SELECT SUM(total) FROM customer WHERE date ='"+sqlDate+"'";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(sql);

result = prepare.executeQuery();

while(result.next()){

incomeToday = result.getDouble("SUM(total)");

}

dashboard\_incomeToday.setText("$"+String.valueOf(incomeToday));

}catch(Exception e){

e.printStackTrace();

}

}

private double totalIncome;

public void dashboardDisplayTI(){

String sql = "SELECT SUM(total) FROM customer";

connect = database.connectDb();

try{

prepare = connect.prepareStatement(sql);

result = prepare.executeQuery();

while(result.next()){

totalIncome = result.getDouble("SUM(total)");

}

dashboard\_totalIncome.setText("$"+String.valueOf(totalIncome));

}catch(Exception e){e.printStackTrace();}

}

public void dashboardChart(){

dashboard\_chart.getData().clear();

String sql = "SELECT date,SUM(total) FROM customer WHERE date != '' GROUP BY date ORDER BY TIMESTAMP(date) ASC LIMIT 9";

connect = database.connectDb();

XYChart.Series chart = new XYChart.Series();

try{

prepare = connect.prepareStatement(sql);

result = prepare.executeQuery();

while(result.next()){

chart.getData().add(new XYChart.Data(result.getString(1), result.getInt(2)));

}

dashboard\_chart.getData().add(chart);

}catch(Exception e){e.printStackTrace();}

}

public void displayUsername(){

username.setText(getData.username);

}

public void close() {

System.exit(0);

}

//NOW LET'S PROCEED TO OUR CHART

public void minimize() {

Stage stage = (Stage) main\_form.getScene().getWindow();

stage.setIconified(true);

}

@Override

public void initialize(URL location, ResourceBundle resources) {

defaultBtn();

displayUsername();

dashboardDisplayAB();

dashboardDisplayIT();

dashboardDisplayTI();

dashboardChart();

comboBoxStatus();

availableBShowBusData();

busIdList();

LocationList();

typeList();

ticketNumList();

genderList();

customersShowDataList();

    }

}

**FXML Document Controller:**

package busticketbookingmanagementsystem;

import java.net.URL;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.AnchorPane;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

public class FXMLDocumentController implements Initializable {

@FXML

private AnchorPane main\_form;

@FXML

private TextField username;

@FXML

private PasswordField password;

@FXML

private Button login;

@FXML

private Button close;

// DATABASE TOOLS

private Connection connect;

private PreparedStatement prepare;

private ResultSet result;

private double x = 0;

private double y = 0;

public void login() {

String sql = "SELECT \* FROM admin WHERE username = ? and password = ?";

connect = database.connectDb();

Alert alert;

try {

if (username.getText().isEmpty() || password.getText().isEmpty()) {

alert = new Alert(AlertType.ERROR);

alert.setTitle("Ërror Message");

alert.setHeaderText(null);

alert.setContentText("Please fill all blank fields");

alert.showAndWait();

} else {

prepare = connect.prepareStatement(sql);

prepare.setString(1, username.getText());

prepare.setString(2, password.getText());

result = prepare.executeQuery();

if (result.next()) {

getData.username = username.getText();

// THEN PROCEED TO DASHBOARD FORM :

alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Information Message");

alert.setHeaderText(null);

alert.setContentText("Successfully Login!");

alert.showAndWait();

login.getScene().getWindow().hide();

Parent root = FXMLLoader.load(getClass().getResource("dashboard.fxml"));

Stage stage = new Stage();

Scene scene = new Scene(root);

root.setOnMousePressed((MouseEvent event) -> {

x = event.getSceneX();

y = event.getSceneY();

});

root.setOnMouseDragged((MouseEvent event) -> {

stage.setX(event.getScreenX() - x);

stage.setY(event.getScreenY() - y);

});

stage.initStyle(StageStyle.TRANSPARENT);

stage.setScene(scene);

stage.show();

} else {

// IF INCORRECT THE DATA YOU GAVE

alert = new Alert(AlertType.ERROR);

alert.setTitle("Error Message");

alert.setHeaderText(null);

alert.setContentText("Wrong Username/Password");

alert.showAndWait();

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void close() {

System.exit(0);

}

@Override

public void initialize(URL url, ResourceBundle rb) {

 // TODO

    }

}

**Get Data:**

package busticketbookingmanagementsystem;

public class getData {

public static Integer number;

public static String username;

}