

COMSATS UNSIVERSITY ISLAMABAD, LAHORE CAMPUS

Theory Assignment No 3

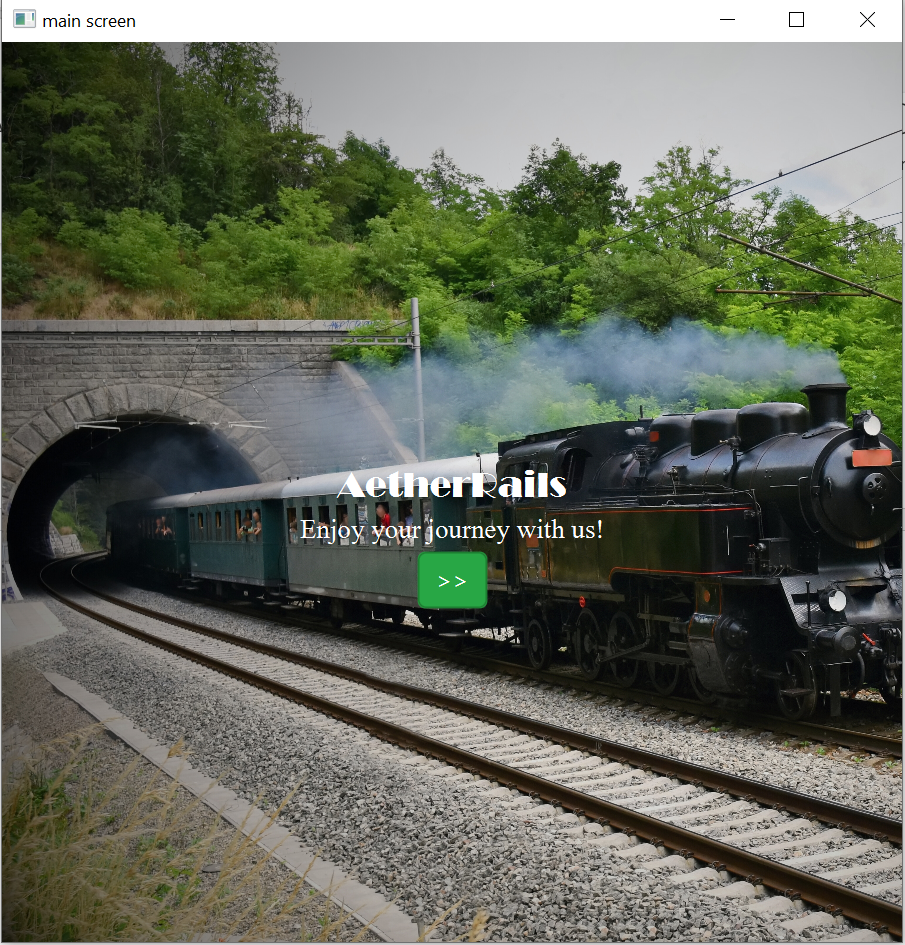
Lab Assignment No 4

Submitted to Sir Shahid Bhatti

Submitted by Haniya Ubaid(SP24-BSE-038)

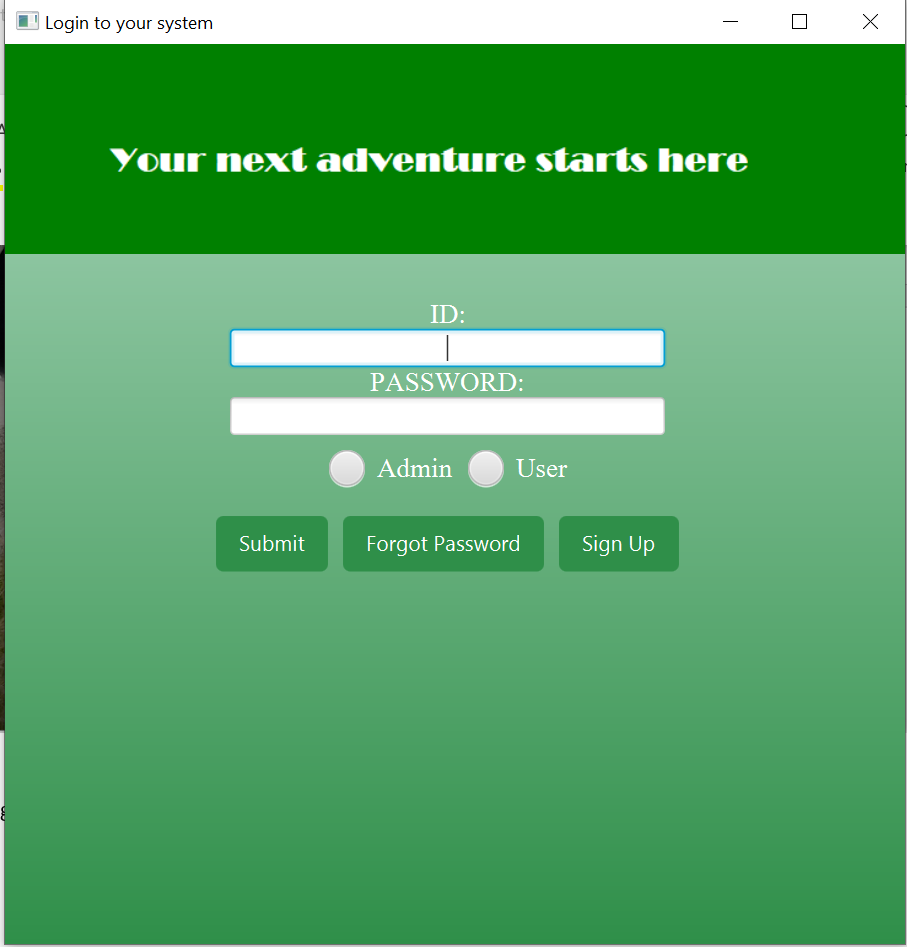
Asma Aslam(SP24-BSE-015)

GUI:



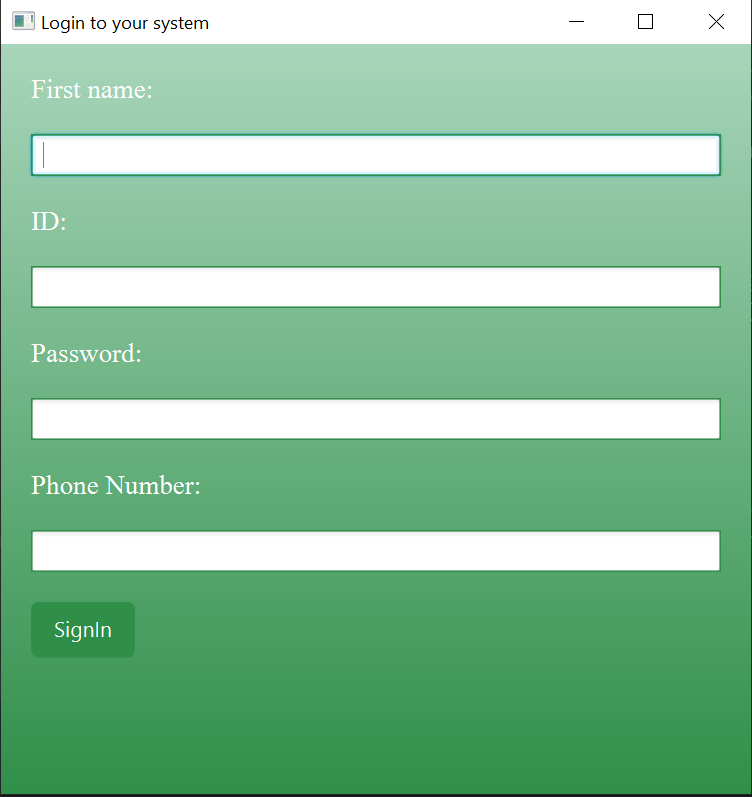
This is the Welcome Page which contains the background picture and label and proceeding button.

Login page

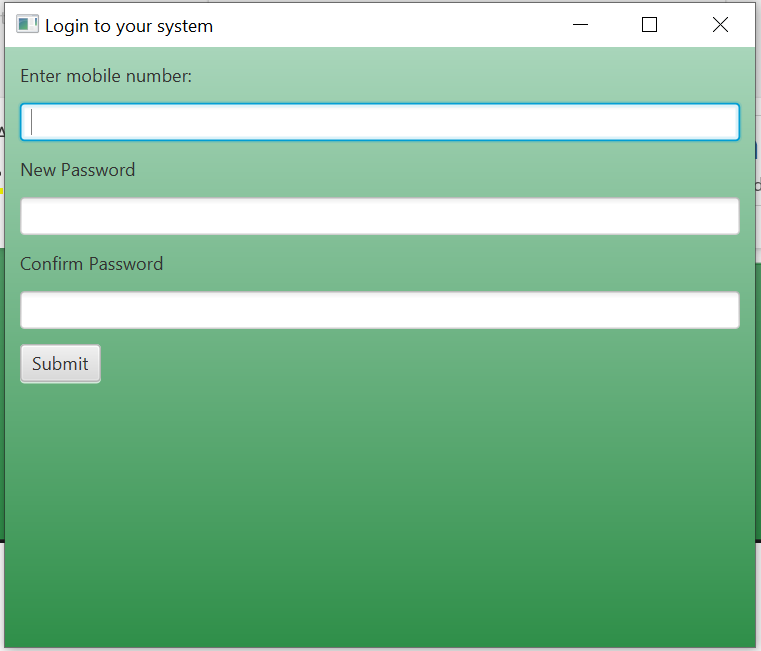


This is the login page which contains textfield and passwordfield that take id and password of the user or admin. This also contains the toggle button in which we select admin and user. It also include submit , Forgot password and Sign up buttons.

SignIn Page:

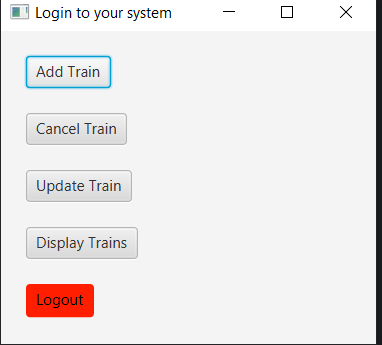
  
This is the SignIn page it takes the data from the textfields and creates your account when you push the sign in button along with that it takes you to the main page.

Forgot Password:



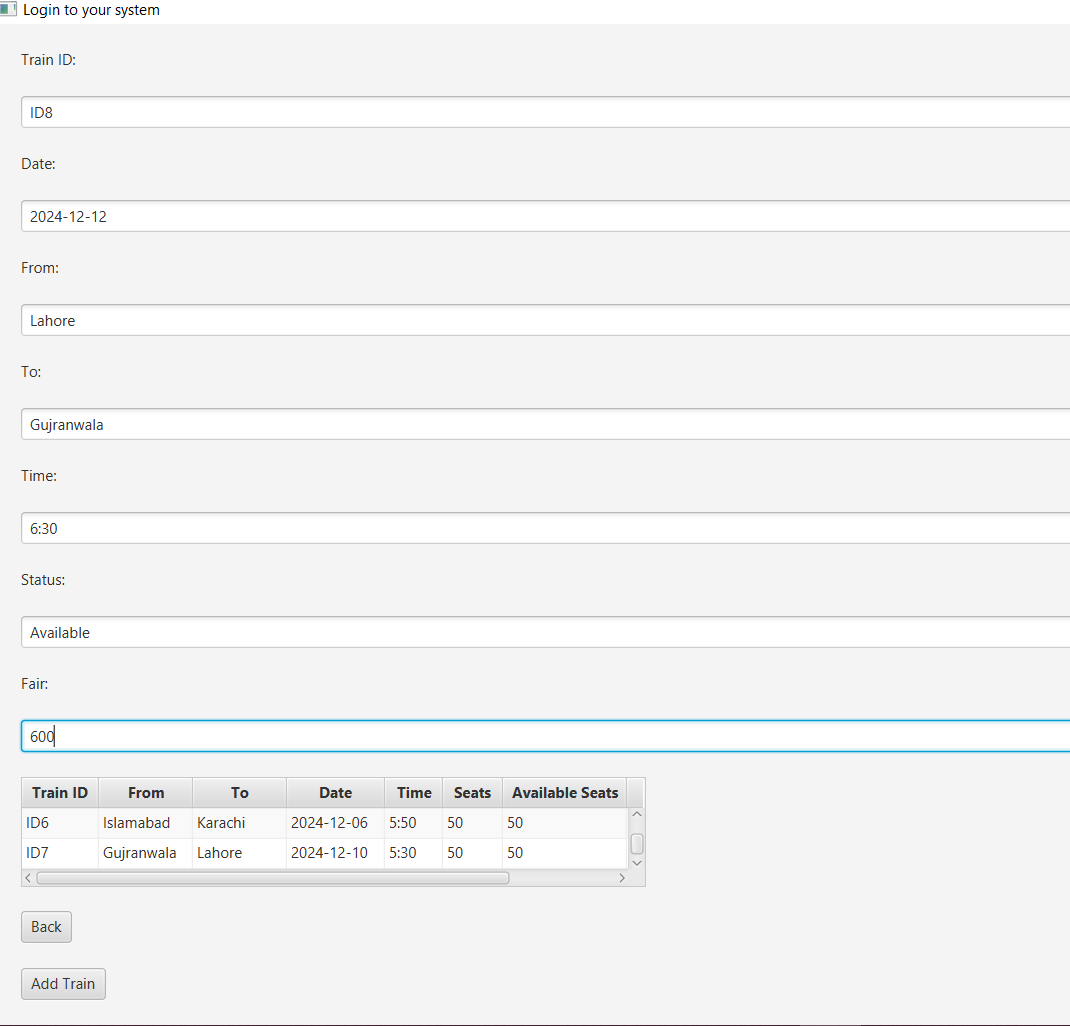
This page is for creating new password it takes the phone number as to validate that the user has an account along with that it takes the new password and confirms it too then submit button makes changes in the account.

Admin Site



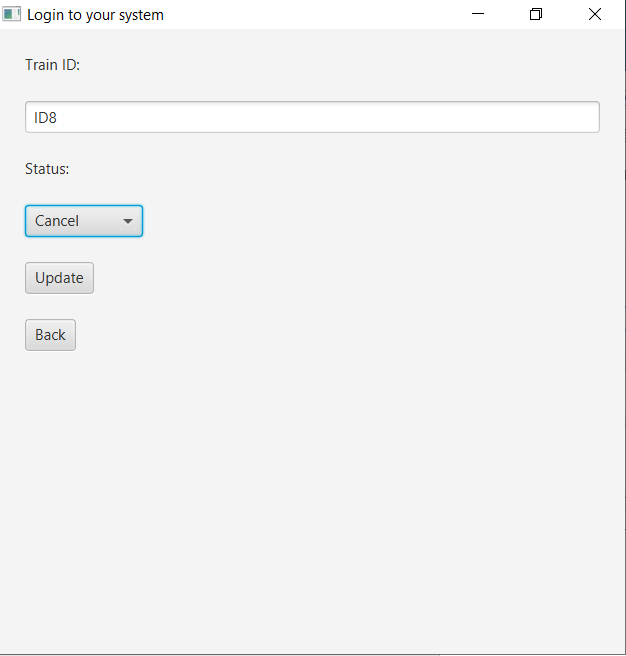
This is the main page for the admin side it contains the buttons of add train, cancel train ,update train ,display train and then lastly logout page which takes you to the main page.

Add Train



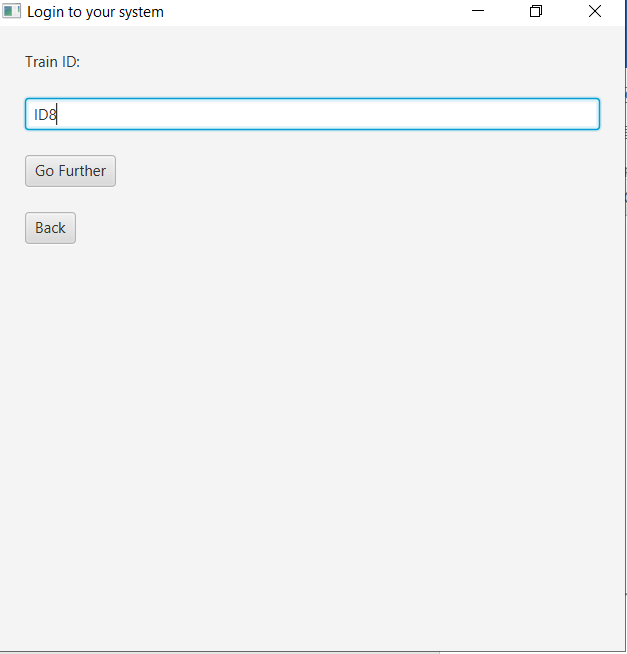
This page takes data from the textfield then it adds your new train to the list when add train button is called it also contains the back button which takes you to the admin page.

Cancel Train

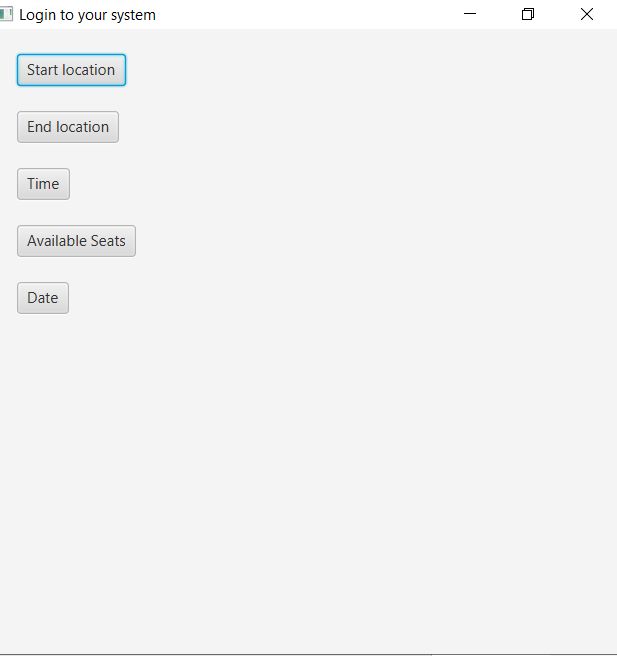


Cancel train its you to this page where you can add the train id which needs to be cancelled you use the combo box to choose what change is needed the the update button makes such change while the back takes you to the admin page.

Update train

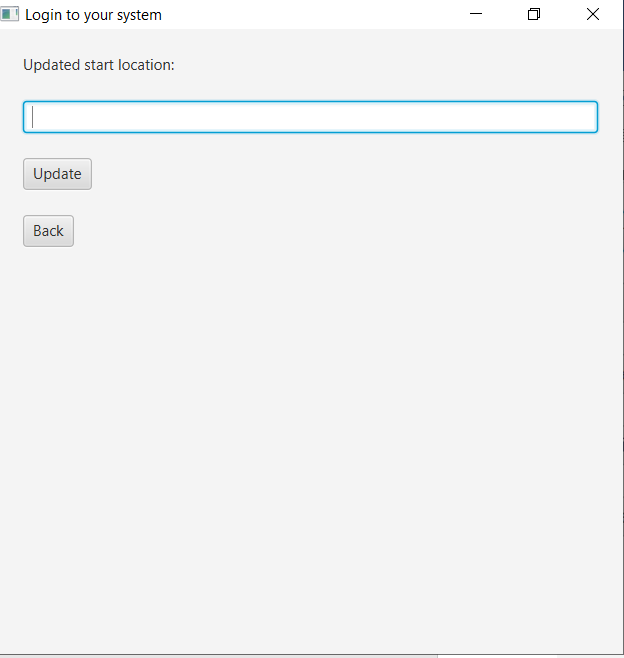


Update train takes the ID of the bus you need to make changes to then proceed further takes you to the next page while back button takes you to the previous page



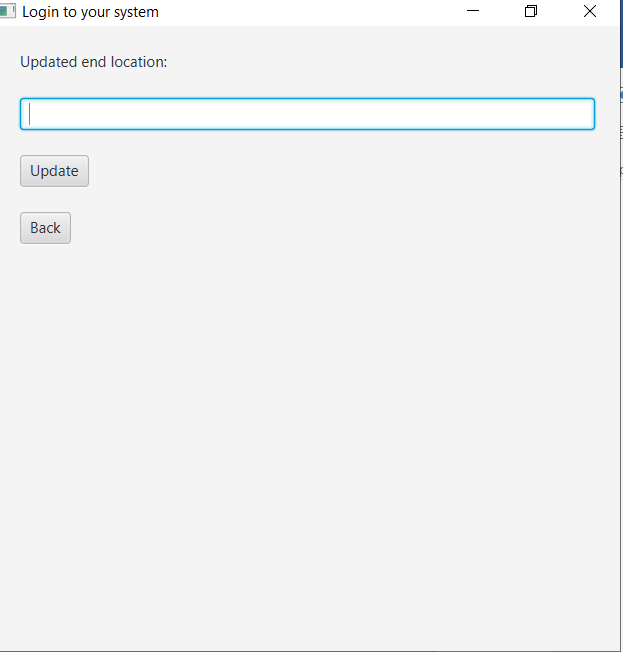
Updation page:

It has the buttons for what change or updation is to be made like what do the admin wants to change the start location, end location , date, time or available seats then the back button takes you to the previous page



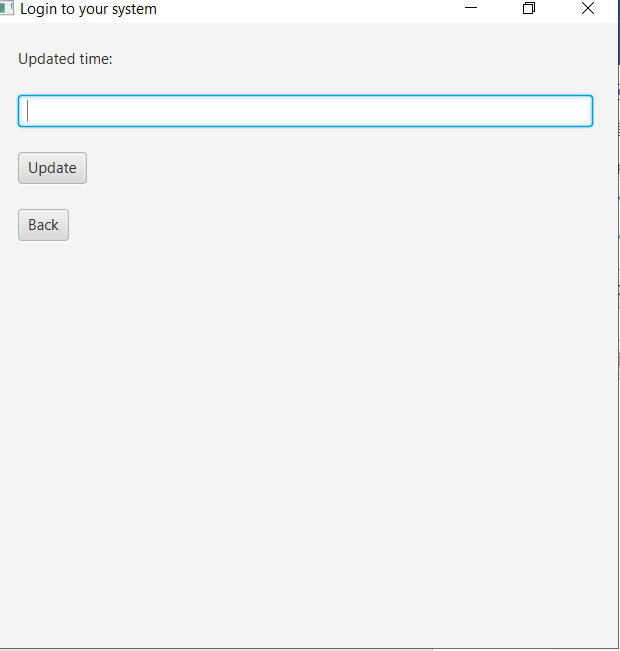
Update Start Location

It takes the data from the textfield and updates what the new info is by clicking the updation button.



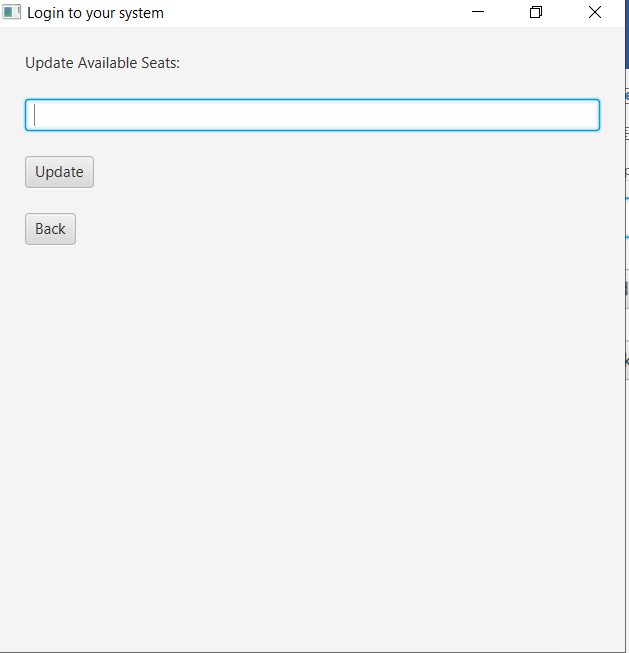
Update End location

It takes the data from the textfield and updates what the new info is by clicking the updation button.



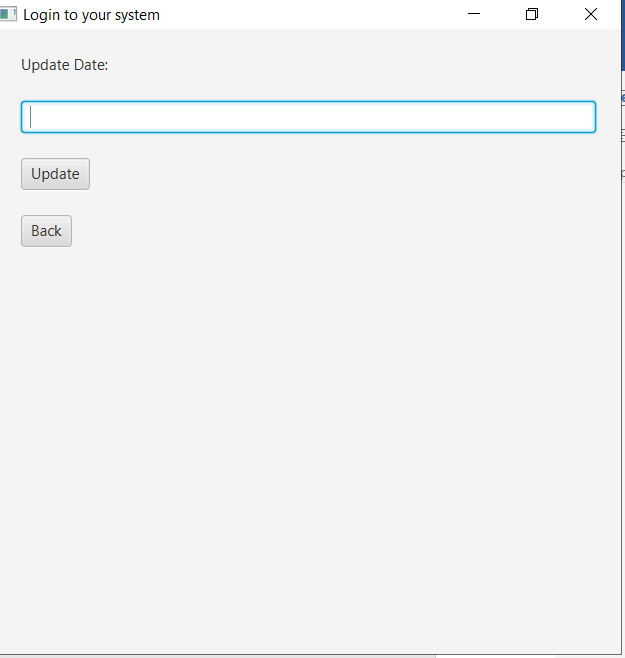
Update Time

It takes the data from the textfield and updates what the new info is by clicking the updation button.



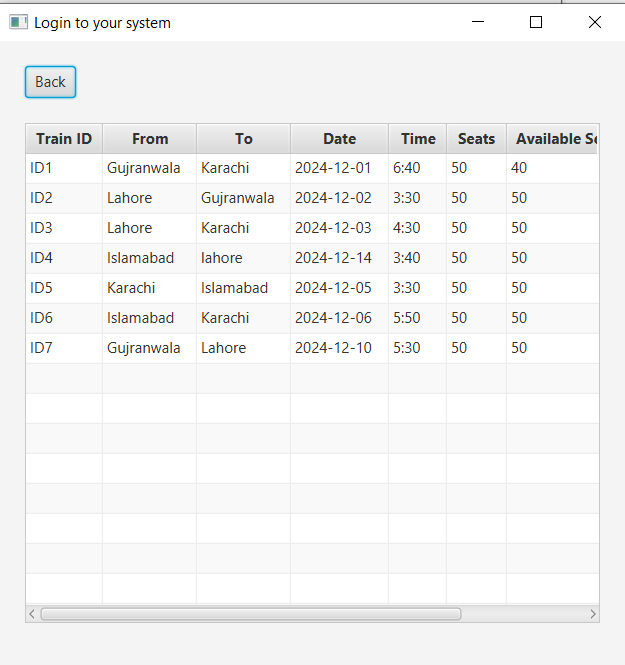
Update Available Seats

It takes the data from the textfield and updates what the new info is by clicking the updation button.



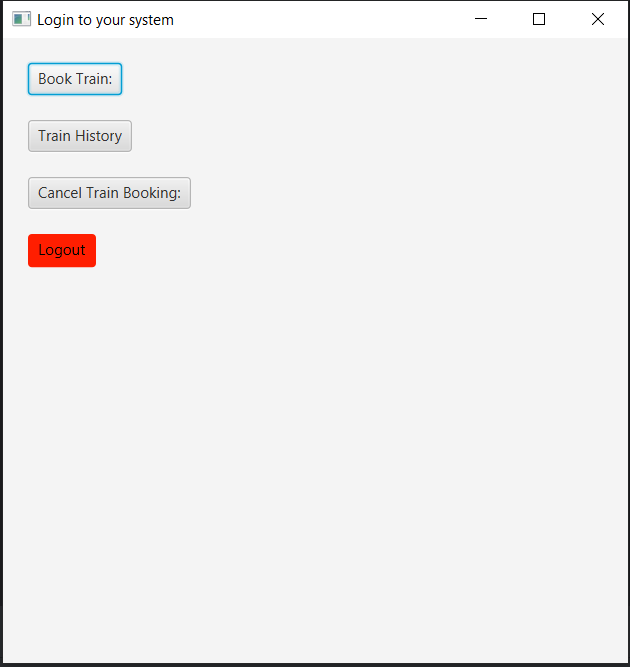
Update Date:

It takes the data from the textfield and updates what the new info is by clicking the updation button.



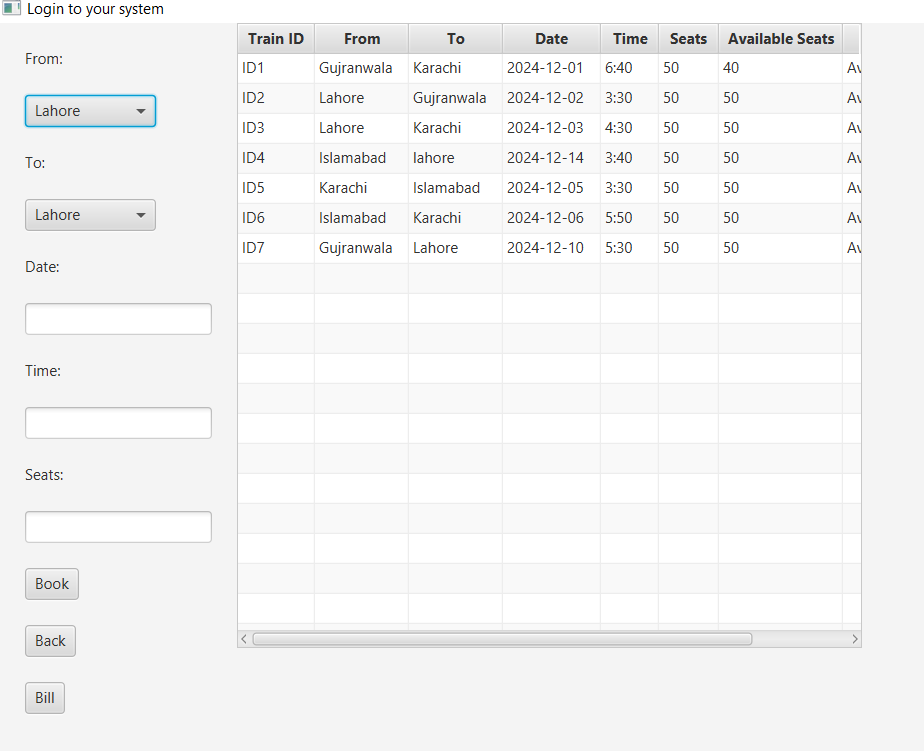
Displays the Train history:

Gives the history of the trains that are going.



User interface:

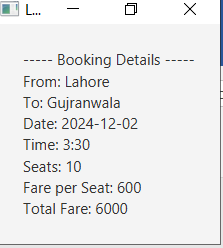
This is the interface that helps you to use the user functions like to book train , display their history, logout and cancel the booking.



Booking the Bus:

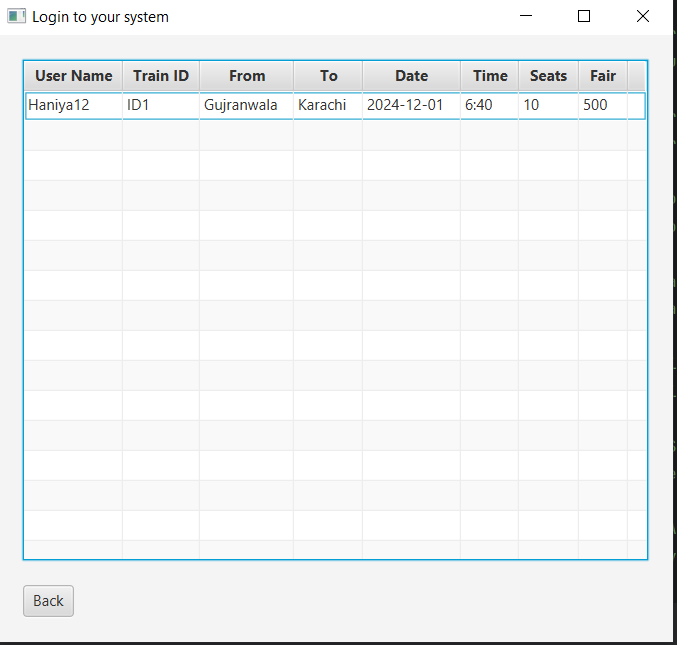
This takes the data from the textfields and then books your ticket while back button takes you to the previous page.

Bill Generate



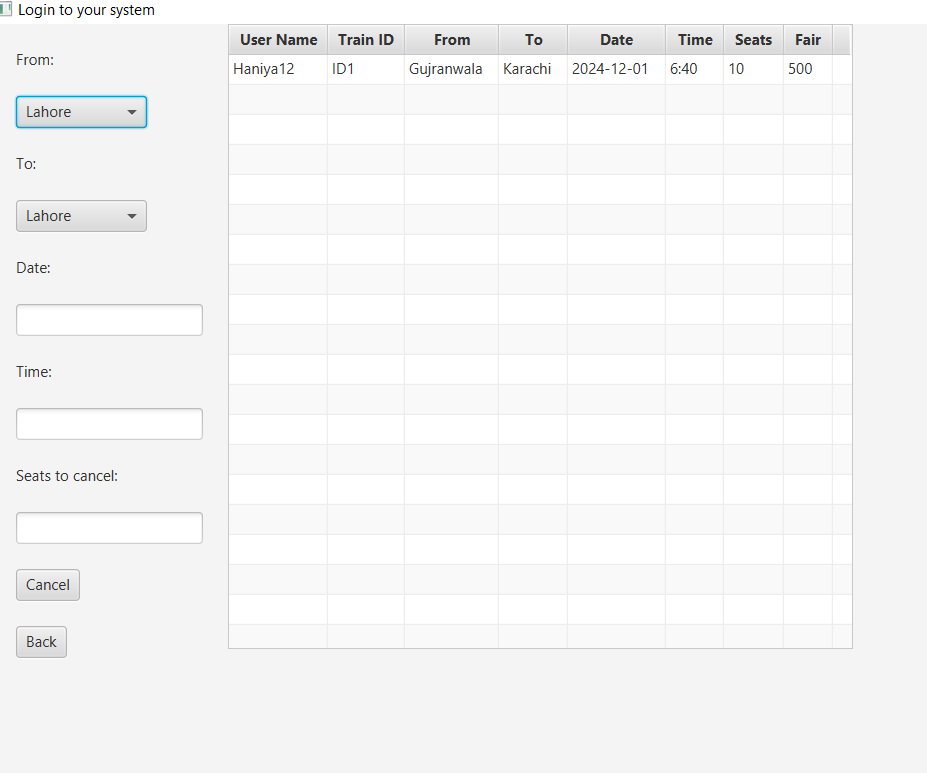
When bill button is clicked it generates your bill by calculating your fair.

Booking History



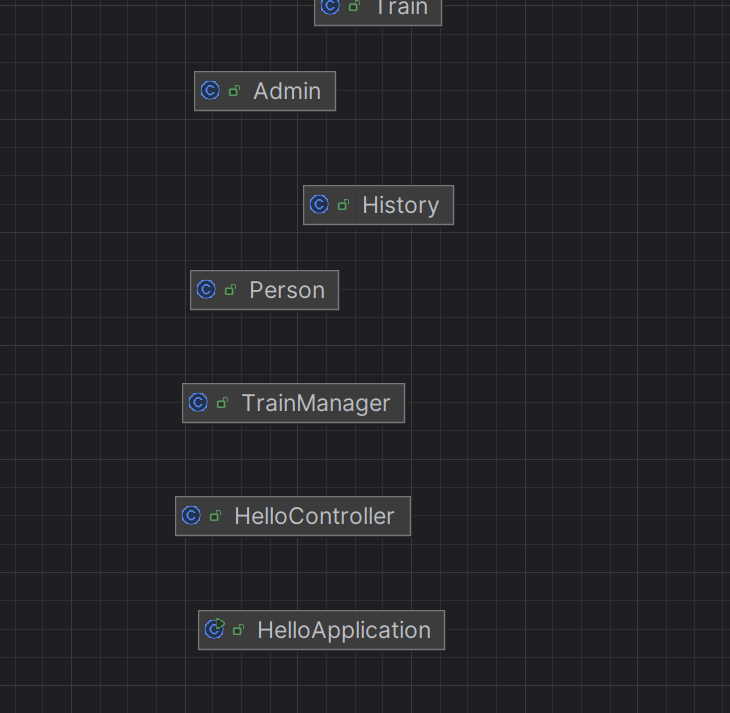
This shows you your previous train history

Cancel Booking



Here you can cancel the train by giving the data and entering how many seats you want to remove

Code



CLASS: HelloApplication

package com.example.oopfinalproject;  
  
import javafx.application.Application;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class HelloApplication extends Application {  
  
 private Scene scene;  
 private Scene s1;  
 private Scene s2;  
 private Scene s3;  
 ObservableList<Train> busm;  
  
  
 @Override  
 public void start(Stage stage) throws IOException {  
 File file = new File("target/generated-sources/annotations/Data");  
 File file1 = new File("target/generated-sources/annotations/Admin");  
 File file2 = new File("target/generated-sources/annotations/Busdetail");  
 File file3 = new File("target/generated-sources/annotations/Busbooking");  
 TrainManager busmanager = new TrainManager();  
 busm = FXCollections.*observableArrayList*(  
 busmanager.readfromfile(file2)  
 );  
  
 ObservableList<Person> list = FXCollections.*observableArrayList*();  
 ObservableList<Admin> list2 = FXCollections.*observableArrayList*();  
  
  
 RadioButton adminRadio = new RadioButton("Admin");  
 RadioButton userRadio = new RadioButton("User");  
 ToggleGroup Group = new ToggleGroup();  
 adminRadio.setToggleGroup(Group);  
 userRadio.setToggleGroup(Group);  
 HBox selectionBox = new HBox(10, adminRadio, userRadio);  
  
  
 // sign up button  
 Button b3 = new Button("Sign Up");  
 b3.setOnAction(e -> {  
 VBox vbox = new VBox();  
 vbox.setPadding(new Insets(20, 20, 20, 20));  
 vbox.setSpacing(20);  
 TextField username = new TextField();  
 Label l = new Label("First name:");  
 TextField ID = new TextField();  
 Label l1 = new Label("Username:");  
 PasswordField Password = new PasswordField();  
 Label l2 = new Label("Password:");  
 TextField Number = new TextField();  
 Label l3 = new Label("Phone Number:");  
  
 Button b4 = new Button("SignIn");  
 if (Group.getSelectedToggle().equals(userRadio)) {  
 b4.setOnAction(submitevent -> {  
 String name = username.getText();  
 String id = ID.getText();  
 String password = Password.getText();  
 String number = Number.getText();  
 *write*(file, name, password, number, id);  
 // list.add(new Person(name, id, password, number));  
 stage.setScene(scene);  
 stage.show();  
 });  
 } else if (Group.getSelectedToggle().equals(adminRadio)) {  
 b4.setOnAction(submitevent -> {  
 String name = username.getText();  
 String id = ID.getText();  
 String password = Password.getText();  
 String number = Number.getText();  
 *write*(file1, name, password, number, id);  
 //list2.add(new Admin(name, id, password, number));  
 stage.setScene(scene);  
 stage.show();  
 });  
 }  
  
 vbox.getChildren().addAll(l, username, l1, ID, l2, Password, l3, Number, b4);  
  
 Scene s1 = new Scene(vbox, 500, 500);  
 stage.setScene(s1);  
 stage.show();  
 username.clear();  
 ID.clear();  
 Password.clear();  
 Number.clear();  
 });  
  
 TextField nameField = new TextField();  
 Label IDlabel = new Label("Username:");  
 PasswordField PasswordField = new PasswordField();  
 Label passwordlabel = new Label("PASSWORD:");  
 VBox vbox = new VBox();  
 vbox.setPadding(new Insets(10, 10, 10, 10));  
 vbox.setSpacing(10);  
  
  
  
 // Submit button  
 Button b1 = new Button("Submit");  
 b1.setOnAction(e -> {  
 String ID = nameField.getText();  
 String password = PasswordField.getText();  
 //User block  
 if (Group.getSelectedToggle().equals(userRadio)) {  
 if (*verify*(file, password, ID)) {  
 System.*out*.println("Username exist");  
 } else {  
  
 System.*out*.println("Username doesnot exist");  
 }  
 nameField.clear();  
 PasswordField.clear();  
  
 TableView<Train> t1 = new TableView<>();  
 t1.setMaxWidth(500);  
 t1.setMaxHeight(500);  
 t1.setEditable(true);  
  
 TableColumn<Train, String> col1 = new TableColumn<>("Train ID");  
 col1.setCellValueFactory(new PropertyValueFactory<>("busId"));  
  
 TableColumn<Train, String> col2 = new TableColumn<>("From");  
 col2.setCellValueFactory(new PropertyValueFactory<>("From"));  
  
 TableColumn<Train, String> col3 = new TableColumn<>("To");  
 col3.setCellValueFactory(new PropertyValueFactory<>("To"));  
  
 TableColumn<Train, String> col4 = new TableColumn<>("Date");  
 col4.setCellValueFactory(new PropertyValueFactory<>("Date"));  
  
 TableColumn<Train, String> col5 = new TableColumn<>("Time");  
 col5.setCellValueFactory(new PropertyValueFactory<>("Time"));  
  
 TableColumn<Train, Integer> col6 = new TableColumn<>("Seats");  
 col6.setCellValueFactory(new PropertyValueFactory<>("seats"));  
  
 TableColumn<Train, Integer> col7 = new TableColumn<>("Available Seats");  
 col7.setCellValueFactory(new PropertyValueFactory<>("availableSeats"));  
  
  
 TableColumn<Train, String> col8 = new TableColumn<>("Status");  
 col8.setCellValueFactory(new PropertyValueFactory<>("status"));  
  
 TableColumn<Train, String> col9 = new TableColumn<>("Fair");  
 col9.setCellValueFactory(new PropertyValueFactory<>("fair"));  
  
 t1.getColumns().addAll(col1, col2, col3, col4, col5, col6, col7, col8, col9);  
 t1.setItems(busmanager.readfromfile(file2));  
  
 VBox vbox1 = new VBox();  
 vbox1.setPadding(new Insets(20, 20, 20, 20));  
 vbox1.setSpacing(20);  
  
 Button logout = new Button("Logout");  
 logout.setStyle("-fx-background-color: #ff1e00;-fx-text-fill: black;");  
 logout.setOnAction(submitresponse -> {  
 stage.setScene(scene);  
 stage.show();  
 });  
  
  
 TableView<History> t2 = new TableView<>();  
 t2.setMaxWidth(500);  
 t2.setMaxHeight(500);  
 t2.setEditable(true);  
  
  
 TableColumn<History, String> co1 = new TableColumn<>("User Name");  
 co1.setCellValueFactory(new PropertyValueFactory<>("Username"));  
  
  
 TableColumn<History, String> co2 = new TableColumn<>("Train ID");  
 co2.setCellValueFactory(new PropertyValueFactory<>("trainid"));  
  
 TableColumn<History, String> co3 = new TableColumn<>("From");  
 co3.setCellValueFactory(new PropertyValueFactory<>("from"));  
  
 TableColumn<History, String> co4 = new TableColumn<>("To");  
 co4.setCellValueFactory(new PropertyValueFactory<>("to"));  
  
 TableColumn<History, String> co5 = new TableColumn<>("Date");  
 co5.setCellValueFactory(new PropertyValueFactory<>("date"));  
  
 TableColumn<History, String> co6 = new TableColumn<>("Time");  
 co6.setCellValueFactory(new PropertyValueFactory<>("time"));  
  
 TableColumn<History, Integer> co7 = new TableColumn<>("Seats");  
 co7.setCellValueFactory(new PropertyValueFactory<>("seats"));  
  
  
 TableColumn<History, String> co8 = new TableColumn<>("Fair");  
 co8.setCellValueFactory(new PropertyValueFactory<>("fair"));  
  
 t2.getColumns().addAll(co1, co2, co3, co4, co5, co6, co7, co8);  
  
 Button history = new Button("Train History");  
 history.setOnAction(submitevent -> {  
  
 VBox vbox2 = new VBox();  
 vbox2.setPadding(new Insets(20, 20, 20, 20));  
 vbox2.setSpacing(20);  
 Button back = new Button("Back");  
 back.setOnAction(k -> {  
 stage.setScene(s2);  
 stage.show();  
  
 });  
  
 t2.setItems(historylist(file3, ID));  
 vbox2.getChildren().addAll(t2, back);  
 Scene scene = new Scene(vbox2);  
 stage.setScene(scene);  
 stage.show();  
  
 });  
  
 Button cancel = new Button("Cancel Train Booking:");  
 cancel.setOnAction(submitevent -> {  
  
 VBox vbox2 = new VBox();  
 vbox2.setPadding(new Insets(20, 20, 20, 20));  
 vbox2.setSpacing(20);  
 ComboBox<String> fromComboBox = new ComboBox<>();  
 fromComboBox.getItems().addAll("Lahore", "Karachi", "Islamabad", "Gujranwala");  
 fromComboBox.setValue("Lahore");  
 Label l = new Label("From:");  
  
  
 Label l1 = new Label("To:");  
 ComboBox<String> toComboBox = new ComboBox<>();  
 toComboBox.getItems().addAll("Lahore", "Karachi", "Islamabad", "Gujranwala");  
 toComboBox.setValue("Lahore");  
  
  
 Label l2 = new Label("Date:");  
 TextField d1 = new TextField();  
 Label l3 = new Label("Time:");  
 TextField time = new TextField();  
 Label l4 = new Label("Seats to cancel:");  
 TextField seats = new TextField();  
 Button book1 = new Button("Cancel");  
  
  
 book1.setOnAction(w -> {  
 String from = fromComboBox.getValue();  
 String to = toComboBox.getValue();  
 String date = d1.getText();  
 String time1 = time.getText();  
 String seats1 = seats.getText();  
 String trainID = verifybusinfo(from, to, date, time1, file2);  
 busmanager.*Cancelseats*(trainID, seats1, file2);  
 canceltrainbooking(file3, ID, trainID,seats1);  
 d1.clear();  
 time.clear();  
 seats.clear();  
  
  
  
  
 });  
 Button back = new Button("Back");  
 back.setOnAction(q -> {  
 stage.setScene(s2);  
 stage.show();  
 });  
 HBox hbox = new HBox();  
  
 t2.setItems(historylist(file3, ID));  
 vbox2.getChildren().addAll(l, fromComboBox, l1, toComboBox, l2, d1, l3, time, l4, seats, book1, back);  
 hbox.getChildren().addAll(vbox2, t2);  
 Scene c1 = new Scene(hbox, 700, 500);  
 stage.setScene(c1);  
 stage.show();  
  
  
 });  
  
  
 Button book = new Button("Book Train:");  
 book.setOnAction(submitevent -> {  
  
 VBox vbox2 = new VBox();  
 vbox2.setPadding(new Insets(20, 20, 20, 20));  
 vbox2.setSpacing(20);  
 ComboBox<String> fromComboBox = new ComboBox<>();  
 fromComboBox.getItems().addAll("Lahore", "Karachi", "Islamabad", "Gujranwala");  
 fromComboBox.setValue("Lahore");  
 Label l = new Label("From:");  
  
  
 Label l1 = new Label("To:");  
 ComboBox<String> toComboBox = new ComboBox<>();  
 toComboBox.getItems().addAll("Lahore", "Karachi", "Islamabad", "Gujranwala");  
 toComboBox.setValue("Lahore");  
  
  
 Label l2 = new Label("Date:");  
 TextField d1 = new TextField();  
 Label l3 = new Label("Time:");  
 TextField time = new TextField();  
 Label l4 = new Label("Seats:");  
 TextField seats = new TextField();  
 Button book1 = new Button("Book");  
  
 book1.setOnAction(w -> {  
 String from = fromComboBox.getValue();  
 String to = toComboBox.getValue();  
 String date = d1.getText();  
 String time1 = time.getText();  
 String seats1 = seats.getText();  
 String trainID = verifybusinfo(from, to, date, time1, file2);  
 String trainfair = trainfair(from, to, date, time1, file2);  
  
  
 busmanager.*Availseats*(trainID, seats1, file2);  
 historytrainbooking(file2, file3, ID, trainID, seats1);  
  
  
 });  
  
  
  
 Button bill = new Button("Bill");  
 bill.setOnAction(q -> {  
 VBox vbox3 = new VBox();  
 vbox3.setPadding(new Insets(20, 20, 20, 20));  
 vbox3.setSpacing(20);  
 String from = fromComboBox.getValue();  
 String to = toComboBox.getValue();  
 String date = d1.getText();  
 String time1 = time.getText();  
 String seats1 = seats.getText();  
 String busID = verifybusinfo(from, to, date, time1, file2);  
 String busfair = trainfair(from, to, date, time1, file2);  
 int seatCount = Integer.*parseInt*(seats1);  
 int farePerSeat = Integer.*parseInt*(busfair);  
 int totalFare = seatCount \* farePerSeat;  
  
 String billslip = "----- Booking Details -----\n" +  
 "From: " + from + "\n" +  
 "To: " + to + "\n" +  
 "Date: " + date + "\n" +  
 "Time: " + time1 + "\n" +  
 "Seats: " + seats1 + "\n" +  
 "Fare per Seat: " + farePerSeat + "\n" +  
 "Total Fare: " + totalFare + "\n";  
 Label label = new Label(billslip);  
 vbox3.getChildren().add(label);  
  
  
 Scene scene1 = new Scene(vbox3);  
 stage.setScene(scene1);  
 stage.show();  
  
 });  
  
  
 Button back = new Button("Back");  
 back.setOnAction(q -> {  
 stage.setScene(s2);  
 stage.show();  
 });  
 HBox hbox = new HBox();  
  
  
 vbox2.getChildren().addAll(l, fromComboBox, l1, toComboBox, l2, d1, l3, time, l4, seats, book1, back, bill);  
 hbox.getChildren().addAll(vbox2, t1);  
 Scene s3 = new Scene(hbox, 700, 500);  
 stage.setScene(s3);  
 stage.show();  
  
 });  
  
  
 vbox1.getChildren().addAll(book, history, cancel, logout);  
  
 s2 = new Scene(vbox1, 500, 500);  
 stage.setScene(s2);  
 stage.show();  
 } //Admin block  
 else if (Group.getSelectedToggle().equals(adminRadio)) {  
 if (*verify*(file1, password, ID)) {  
 } else {  
 System.*out*.println("This ID does not exist or password is incorrect");  
 }  
 //Table  
 TableView<Train> t1 = new TableView<>();  
 t1.setMaxWidth(500);  
 t1.setMaxHeight(500);  
 t1.setEditable(true);  
  
 TableColumn<Train, String> col1 = new TableColumn<>("Train ID");  
 col1.setCellValueFactory(new PropertyValueFactory<>("busId"));  
  
 TableColumn<Train, String> col2 = new TableColumn<>("From");  
 col2.setCellValueFactory(new PropertyValueFactory<>("From"));  
  
 TableColumn<Train, String> col3 = new TableColumn<>("To");  
 col3.setCellValueFactory(new PropertyValueFactory<>("To"));  
  
 TableColumn<Train, String> col4 = new TableColumn<>("Date");  
 col4.setCellValueFactory(new PropertyValueFactory<>("Date"));  
  
 TableColumn<Train, String> col5 = new TableColumn<>("Time");  
 col5.setCellValueFactory(new PropertyValueFactory<>("Time"));  
  
 TableColumn<Train, Integer> col6 = new TableColumn<>("Seats");  
 col6.setCellValueFactory(new PropertyValueFactory<>("seats"));  
  
 TableColumn<Train, Integer> col7 = new TableColumn<>("Available Seats");  
 col7.setCellValueFactory(new PropertyValueFactory<>("availableSeats"));  
  
  
 TableColumn<Train, String> col8 = new TableColumn<>("Status");  
 col8.setCellValueFactory(new PropertyValueFactory<>("status"));  
  
 TableColumn<Train, String> col9 = new TableColumn<>("Fair");  
 col9.setCellValueFactory(new PropertyValueFactory<>("fair"));  
  
 t1.getColumns().addAll(col1, col2, col3, col4, col5, col6, col7, col8, col9);  
 t1.setItems(busmanager.readfromfile(file2));  
  
 //add button action  
 Button add = new Button("Add Train");  
 add.setOnAction(event -> {  
 VBox vbox1 = new VBox();  
 vbox1.setPadding(new Insets(20, 20, 20, 20));  
 vbox1.setSpacing(20);  
 TextField bus = new TextField();  
 Label l = new Label("Train ID:");  
 TextField tf1 = new TextField();  
 Label l1 = new Label("Date:");  
 TextField tf2 = new TextField();  
 Label l2 = new Label("From:");  
 TextField tf3 = new TextField();  
 Label l3 = new Label("To:");  
 TextField tf4 = new TextField();  
 Label l4 = new Label("Time:");  
 Label l5 = new Label("Status:");  
 TextField status = new TextField();  
 Label l6 = new Label("Fair:");  
 TextField fair = new TextField();  
 Button add1 = new Button("Add Train");  
  
 //Adding in the list  
 add1.setOnAction(p -> {  
 String id = bus.getText();  
 String Date = tf1.getText();  
 String From = tf2.getText();  
 String To = tf3.getText();  
 String Time = tf4.getText();  
 final int seats = 50;  
 int availableSeats = 50;  
 String s = status.getText();  
 String f = fair.getText();  
  
 Train b = new Train(id, From, To, Date, Time, seats, availableSeats, s, f);  
 busmanager.getAllBuses().add(b);  
  
 writetraindetail(file2, id, From, To, Date, Time, seats, availableSeats, s, f);  
  
 t1.setItems(FXCollections.*observableArrayList*(busmanager.getAllBuses()));  
 t1.refresh();  
  
 // Clear text fields  
 bus.clear();  
 tf1.clear();  
 tf2.clear();  
 tf3.clear();  
 tf4.clear();  
 status.clear();  
 fair.clear();  
 });  
 Button back = new Button("Back");  
 back.setOnAction(submitaction -> {  
 stage.setScene(s1);  
 stage.show();  
  
 });  
  
 vbox1.getChildren().addAll(l, bus, l1, tf1, l2, tf2, l3, tf3, l4, tf4, l5, status, l6, fair, t1, back, add1);  
 Scene s1 = new Scene(vbox1, 500, 500);  
 stage.setScene(s1);  
 stage.show();  
  
 });  
  
  
 // delete button action change it to cancel for now  
 Button delete = new Button("Cancel Train");  
 delete.setOnAction(event -> {  
 VBox vbox2 = new VBox();  
 vbox2.setPadding(new Insets(20, 20, 20, 20));  
 vbox2.setSpacing(20);  
 TextField tf = new TextField();  
 Label l1 = new Label("Train ID:");  
 ComboBox<String> cityComboBox = new ComboBox<>();  
 cityComboBox.getItems().addAll("Available", "Cancel");  
 cityComboBox.setValue("Available");  
 Label cityLabel = new Label("Status:");  
 Button delete1 = new Button("Update");  
 //Delete button action  
 delete1.setOnAction(p -> {  
  
 String Id = tf.getText();  
 String city = cityComboBox.getValue();  
  
  
 busmanager.*updateCancelField*(Id, city, file2);  
  
 });  
 Button back = new Button("Back");  
 back.setOnAction(submitaction -> {  
 stage.setScene(s1);  
 stage.show();  
  
 });  
  
 vbox2.getChildren().addAll(l1, tf, cityLabel, cityComboBox, delete1, back);  
 Scene s2 = new Scene(vbox2, 500, 500);  
 stage.setScene(s2);  
  
  
 });  
  
  
 //update button action  
 Button update = new Button("Update Train");  
 update.setOnAction(event -> {  
 VBox vbox3 = new VBox();  
 vbox3.setPadding(new Insets(20, 20, 20, 20));  
 vbox3.setSpacing(20);  
 TextField tf = new TextField();  
 Label l1 = new Label("Train ID:");  
 String busid = tf.getText();  
 busmanager.updateBus(tf.getText());  
 Button update1 = new Button("Go Further");  
 update1.setOnAction(p -> {  
 String Id = tf.getText();  
 VBox vbox4 = new VBox();  
 vbox4.setPadding(new Insets(20, 20, 20, 20));  
 vbox4.setSpacing(20);  
 //From location chnager  
 Button start = new Button("Start location");  
 start.setOnAction(r -> {  
 VBox vbox5 = new VBox();  
 vbox5.setPadding(new Insets(20, 20, 20, 20));  
 vbox5.setSpacing(20);  
 TextField tf2 = new TextField();  
 Label l2 = new Label("Updated start location:");  
 Button back=new Button("Back");  
 back.setOnAction(w->{  
 stage.setScene(s3);  
 stage.show();  
  
 });  
  
 Button u = new Button("Update");  
 u.setOnAction(w -> {  
 String sl = tf2.getText();  
 busmanager.*updateFromField*(Id, sl, file2);  
  
 });  
 vbox5.getChildren().addAll(l2, tf2, u,back);  
 Scene scene1 = new Scene(vbox5, 500, 500);  
 stage.setScene(scene1);  
 stage.show();  
  
 });  
  
  
 //To location changer  
 Button end = new Button("End location");  
 end.setOnAction(r -> {  
 TextField tf3 = new TextField();  
 Label newlocation = new Label("Updated end location:");  
  
 Button ud = new Button("Update");  
 ud.setOnAction(w -> {  
 String newlo = tf3.getText();  
 busmanager.*updateToField*(Id, newlo, file2);  
  
  
 });  
 Button back=new Button("Back");  
 back.setOnAction(w->{  
 stage.setScene(s3);  
 stage.show();  
  
 });  
 VBox vbox6 = new VBox();  
 vbox6.setPadding(new Insets(20, 20, 20, 20));  
 vbox6.setSpacing(20);  
 vbox6.getChildren().addAll(newlocation, tf3, ud,back);  
 Scene scene2 = new Scene(vbox6, 500, 500);  
 stage.setScene(scene2);  
 stage.show();  
  
  
 });  
  
  
 Button time = new Button("Time");  
 time.setOnAction(r -> {  
 VBox vbox6 = new VBox();  
 vbox6.setPadding(new Insets(20, 20, 20, 20));  
 vbox6.setSpacing(20);  
 TextField tf4 = new TextField();  
 Label newtime = new Label("Updated time:");  
  
 Button upd = new Button("Update");  
 upd.setOnAction(w -> {  
 String newt = tf4.getText();  
 busmanager.*updateTimefield*(Id, newt, file2);  
  
 });  
  
  
 Button back=new Button("Back");  
 back.setOnAction(w->{  
 stage.setScene(s3);  
 stage.show();  
  
 });  
 vbox6.getChildren().addAll(newtime, tf4, upd,back);  
 Scene scene2 = new Scene(vbox6, 500, 500);  
  
 stage.setScene(scene2);  
 stage.show();  
  
  
 });  
  
 Button Availableseats = new Button("Available Seats");  
 Availableseats.setOnAction(r -> {  
 TextField tf4 = new TextField();  
 Label lb = new Label("Update Available Seats:");  
 Button upda = new Button("Update");  
 upda.setOnAction(w -> {  
 String str = tf4.getText();  
 busmanager.*updateAvailseats*(Id, str, file2);  
  
  
 });  
 Button back=new Button("Back");  
 back.setOnAction(w->{  
 stage.setScene(s3);  
 stage.show();  
  
 });  
  
 VBox vbox7 = new VBox();  
 vbox7.setPadding(new Insets(20, 20, 20, 20));  
 vbox7.setSpacing(20);  
 vbox7.getChildren().addAll(lb, tf4, upda,back);  
 Scene scene3 = new Scene(vbox7, 500, 500);  
 stage.setScene(scene3);  
 stage.show();  
  
  
 });  
  
  
 Button date = new Button("Date");  
 date.setOnAction(r -> {  
 TextField tf4 = new TextField();  
 Label lb = new Label("Update Date:");  
 Button upd = new Button("Update");  
 upd.setOnAction(w -> {  
 String str = tf4.getText();  
 busmanager.*updateDateField*(Id, str, file2);  
  
  
 });  
  
 Button back=new Button("Back");  
 back.setOnAction(w->{  
 stage.setScene(s3);  
 stage.show();  
  
 });  
 VBox vbox8 = new VBox();  
 vbox8.setPadding(new Insets(20, 20, 20, 20));  
 vbox8.setSpacing(20);  
 vbox8.getChildren().addAll(lb, tf4, upd,back);  
 Scene scene3 = new Scene(vbox8, 500, 500);  
  
 stage.setScene(scene3);  
 stage.show();  
  
  
 });  
  
  
 vbox4.getChildren().addAll(start, end, time, Availableseats, date);  
 Scene s3 = new Scene(vbox4, 500, 500);  
 stage.setScene(s3);  
 stage.show();  
  
 });  
 Button back = new Button("Back");  
 back.setOnAction(submitaction -> {  
 stage.setScene(s1);  
 stage.show();  
  
 });  
 vbox3.getChildren().addAll(l1, tf, update1, back);  
 s3 = new Scene(vbox3, 500, 500);  
 stage.setScene(s3);  
 stage.show();  
  
  
 });  
  
  
 Button display = new Button("Display Trains");  
 display.setOnAction(event -> {  
 VBox vbox4 = new VBox();  
 vbox4.setPadding(new Insets(20, 20, 20, 20));  
 vbox4.setSpacing(20);  
 Button back = new Button("Back");  
 back.setOnAction(submitaction -> {  
 stage.setScene(s1);  
 stage.show();  
  
 });  
 vbox4.getChildren().addAll(back, t1);  
 Scene s4 = new Scene(vbox4, 500, 500);  
 stage.setScene(s4);  
  
  
 });  
  
  
 Button logout = new Button("Logout");  
 logout.setStyle("-fx-background-color: #ff1e00;-fx-text-fill: black;");  
 logout.setOnAction(submitresponse -> {  
 stage.setScene(scene);  
 stage.show();  
 });  
  
  
 VBox vbox2 = new VBox();  
 vbox2.setPadding(new Insets(20, 20, 20, 20));  
 vbox2.setSpacing(20);  
 vbox2.getChildren().addAll(add, delete, update, display, logout);  
  
 s1 = new Scene(vbox2, 300, 250);  
 stage.setScene(s1);  
 stage.show();  
 }  
 nameField.clear();  
 PasswordField.clear();  
 }  
  
 );  
  
 Button b2 = new Button("Forgot Password");  
 b2.setOnAction(e -> {  
 VBox forgotPasswordLayout = new VBox();  
 forgotPasswordLayout.setPadding(new Insets(10, 10, 10, 10));  
 forgotPasswordLayout.setSpacing(10);  
  
 Label newPasswordLabel = new Label("Enter mobile number:");  
 TextField numberField = new TextField();  
 Label l = new Label("New Password");  
 PasswordField tf = new PasswordField();  
 Label l1 = new Label("Confirm Password");  
 PasswordField tf1 = new PasswordField();  
  
 Button submitNewPasswordButton = new Button("Submit");  
 submitNewPasswordButton.setOnAction(submitEvent -> {  
 if (Group.getSelectedToggle().equals(userRadio)) {  
 if (!tf.getText().equals(tf1.getText())) {  
 System.*out*.println("Passwords do not match");  
 } else {  
 *forgotPassword*(file, numberField.getText(), tf.getText());  
 System.*out*.println("Password updated successfully.");  
 }  
 } else if (Group.getSelectedToggle().equals(adminRadio)) {  
 if (!tf.getText().equals(tf1.getText())) {  
 System.*out*.println("Passwords do not match");  
 } else {  
 *forgotPassword*(file1, numberField.getText(), tf.getText());  
 System.*out*.println("Password updated successfully.");  
 }  
 }  
 });  
 forgotPasswordLayout.getChildren().addAll(newPasswordLabel, numberField, l, tf, l1, tf1, submitNewPasswordButton);  
 Scene forgotPasswordScene = new Scene(forgotPasswordLayout, 500, 400);  
  
 stage.setScene(forgotPasswordScene);  
 stage.show();  
 tf1.clear();  
 tf.clear();  
 numberField.clear();  
 });  
  
 HBox hbox = new HBox();  
 hbox.setPadding(new Insets(10, 10, 10, 10));  
 hbox.setSpacing(10);  
 hbox.getChildren().addAll(b1, b2, b3);  
  
 vbox.getChildren().addAll(IDlabel, nameField, passwordlabel, PasswordField, selectionBox, hbox);  
  
 scene = new Scene(vbox, 300, 250);  
 stage.setTitle("Login to your system");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 private static void write(File file, String name, String password, String number, String id) {  
 try (BufferedWriter bf = new BufferedWriter(new FileWriter(file, true))) {  
 bf.write(id + " " + password + " " + number + " " + name);  
 bf.newLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 private static boolean verify(File file, String password, String ID) {  
 try (BufferedReader bf = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = bf.readLine()) != null) {  
 String[] parts = line.split(" ");  
 if (parts[0].equals(ID) && parts[1].equals(password)) {  
 return true;  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 private static void forgotPassword(File file, String number, String newPassword) {  
 List<String> updatedLines = new ArrayList<>();  
 boolean isUpdated = false;  
  
 try (BufferedReader bf = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = bf.readLine()) != null) {  
 String[] parts = line.split(" ");  
 if (parts[2].equals(number)) {  
 parts[1] = newPassword;  
 isUpdated = true;  
 }  
 updatedLines.add(String.*join*(" ", parts));  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 if (isUpdated) {  
 try (BufferedWriter bf = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 bf.write(updatedLine);  
 bf.newLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 public void writetraindetail(File file, String id, String From, String To, String Date, String Time, int seats, int availableSeats, String status, String fair) {  
 try (BufferedWriter bw = new BufferedWriter(new FileWriter(file, true))) {  
 bw.write(id + " " + From + " " + To + " " + Date + " " + Time + " " + seats + " " + availableSeats + " " + status + " " + fair);  
 bw.newLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void writetrainbookingdetail(File file2, String Username, String id, String from, String to, String date, String time, String seats, String availableSeats, String status, String fair) {  
 try (BufferedWriter bf = new BufferedWriter(new FileWriter(file2, true))) {  
 bf.write(Username + " " + id + " " + from + " " + to + " " + date + " " + time + " " + seats + " " + availableSeats + " " + status + " " + fair);  
 bf.newLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
  
  
 }  
 }  
  
  
 //write trainbooking deatil along with the username  
 public void historytrainbooking(File file1, File file2, String username, String id, String seat) {  
 try (BufferedReader bf = new BufferedReader(new FileReader(file1))) {  
 String line;  
 while ((line = bf.readLine()) != null) {  
 String[] parts = line.split(" ", 9);  
 if (parts[0].trim().equals(id.trim())) {  
 String ID = parts[0];  
 String from = parts[1];  
 String to = parts[2];  
 String date = parts[3];  
 String time = parts[4];  
 String seats = parts[5];  
 String availableSeats = seat;  
 String status = parts[7];  
 String fair = parts[8];  
 writetrainbookingdetail(file2, username, ID, from, to, date, time, seats, availableSeats, status, fair);  
 }  
  
  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public String verifybusinfo(String from, String to, String date, String time, File file) {  
 try (BufferedReader bf = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = bf.readLine()) != null) {  
 String[] parts = line.split(" ", 9);  
  
 if (parts.length >= 5 && parts[1].equals(from) && parts[2].equals(to) && parts[3].equals(date) && parts[4].equals(time)) {  
 String ID = parts[0];  
 return ID;  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public ObservableList<History> historylist(File file,String Username) {  
 ObservableList<History> list = FXCollections.*observableArrayList*();  
 try(BufferedReader bf=new BufferedReader(new FileReader(file))){  
 String line;  
 while((line=bf.readLine())!=null) {  
 String[] parts = line.split(" ");  
 if(parts[0].equals(Username)) {  
 int seats=Integer.*parseInt*(parts[7]);  
 list.add(new History(parts[0],parts[1],parts[2],parts[3],parts[4],parts[5],seats,parts[9]));  
  
 }  
  
 }  
 }  
  
  
 catch(IOException e){  
 e.printStackTrace();  
 }  
  
  
 return list;  
  
 }  
 public String trainfair(String from, String to, String date, String time, File file) {  
 try (BufferedReader bf = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = bf.readLine()) != null) {  
 String[] parts = line.split(" ", 9);  
  
 if (parts.length >= 9 && parts[1].equals(from) && parts[2].equals(to) && parts[3].equals(date) && parts[4].equals(time)) {  
 String fair = parts[8];  
  
 return fair;  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
  
public void canceltrainbooking(File file, String username, String trainID, String cancelseats){  
 boolean busFound = false;  
 String up;  
 List<String> updatedLines = new ArrayList<>();  
 int newnumber = Integer.*parseInt*(cancelseats);  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
 if (parts[0].equals(username)&&parts[1].equals(trainID)) {  
 busFound = true;  
  
 int availableSeats = Integer.*parseInt*(parts[7]);  
 if (availableSeats - newnumber >= 0) {  
 availableSeats -= newnumber;  
 parts[7] = String.*valueOf*(availableSeats);  
 } else {  
 System.*out*.println("No more seats can be cancelled");  
 return;  
 }  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + trainID + " not found in the list.");  
 }  
  
  
  
  
}  
  
  
  
  
  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

Class: Train

package com.example.oopfinalproject;  
  
import javafx.beans.property.SimpleStringProperty;  
  
public class Train {  
 private SimpleStringProperty busId;  
 private SimpleStringProperty From;  
 private SimpleStringProperty To;  
 private SimpleStringProperty Date;  
 private SimpleStringProperty Time;  
 private SimpleStringProperty Status;  
 private SimpleStringProperty Fair;  
 private int seats;  
 private int availableSeats;  
  
  
 public String getStatus() {  
 return Status.get();  
 }  
  
 public SimpleStringProperty statusProperty() {  
 return Status;  
 }  
  
 public void setStatus(String status) {  
 this.Status.set(status);  
 }  
  
 public String getFair() {  
 return Fair.get();  
 }  
  
 public SimpleStringProperty fairProperty() {  
 return Fair;  
 }  
  
 public void setFair(String fair) {  
 this.Fair.set(fair);  
 }  
  
 public Train(String busId, String from, String to, String date, String time, int seats, int availableSeats, String status,  
 String fair) {  
 this.busId = new SimpleStringProperty(busId);  
 this.From = new SimpleStringProperty(from);  
 this.To = new SimpleStringProperty(to);  
 this.Date = new SimpleStringProperty(date);  
 this.Time = new SimpleStringProperty(time);  
 this.seats = seats;  
 this.availableSeats = availableSeats;  
 this.Status = new SimpleStringProperty(status);  
 this.Fair = new SimpleStringProperty(fair);  
 }  
  
  
  
  
 // Getters and setters for each property  
 public String getBusId() {  
 return busId.get();  
 }  
  
 public SimpleStringProperty busIdProperty() {  
 return busId;  
 }  
  
 public void setBusId(String busId) {  
 this.busId.set(busId);  
 }  
  
 public String getFrom() {  
 return From.get();  
 }  
  
 public SimpleStringProperty fromProperty() {  
 return From;  
 }  
  
 public void setFrom(String from) {  
 this.From.set(from);  
 }  
  
 public String getTo() {  
 return To.get();  
 }  
  
 public SimpleStringProperty toProperty() {  
 return To;  
 }  
  
 public void setTo(String to) {  
 this.To.set(to);  
 }  
  
 public String getDate() {  
 return Date.get();  
 }  
  
 public SimpleStringProperty dateProperty() {  
 return Date;  
 }  
  
 public void setDate(String date) {  
 this.Date.set(date);  
 }  
  
 public String getTime() {  
 return Time.get();  
 }  
  
 public SimpleStringProperty timeProperty() {  
 return Time;  
 }  
  
 public void setTime(String time) {  
 this.Time.set(time);  
 }  
  
 public int getSeats() {  
 return seats;  
 }  
  
 public void setSeats(int seats) {  
 this.seats = seats;  
 }  
  
 public int getAvailableSeats() {  
 return availableSeats;  
 }  
  
 public void setAvailableSeats(int availableSeats) {  
 this.availableSeats = availableSeats;  
 }  
  
 @Override  
 public String toString() {  
 return "Bus{" +  
 "busId=" + busId +  
 ", From=" + From +  
 ", To=" + To +  
 ", Date=" + Date +  
 ", Time=" + Time +  
 ", Status=" + Status +  
 ", Fair=" + Fair +  
 ", seats=" + seats +  
 ", availableSeats=" + availableSeats +  
 '}';  
 }  
}

Class: Admin

package com.example.oopfinalproject;  
import javafx.beans.property.SimpleStringProperty;  
  
public class Admin {  
  
 private SimpleStringProperty name;  
 private SimpleStringProperty ID;  
 private SimpleStringProperty Password;  
 private SimpleStringProperty Number;  
  
 public String getName() {  
 return name.get();  
 }  
  
 public SimpleStringProperty nameProperty() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name.set(name);  
 }  
  
 public String getID() {  
 return ID.get();  
 }  
  
 public SimpleStringProperty IDProperty() {  
 return ID;  
 }  
  
 public void setID(String ID) {  
 this.ID.set(ID);  
 }  
  
 public String getPassword() {  
 return Password.get();  
 }  
  
 public SimpleStringProperty passwordProperty() {  
 return Password;  
 }  
  
 public void setPassword(String password) {  
 this.Password.set(password);  
 }  
  
 public String getNumber() {  
 return Number.get();  
 }  
  
 public SimpleStringProperty numberProperty() {  
 return Number;  
 }  
  
 public void setNumber(String number) {  
 this.Number.set(number);  
 }  
  
 @Override  
 public String toString() {  
 return "Admin{" +  
 "name=" + name +  
 ", ID=" + ID +  
 ", Password=" + Password +  
 ", Number=" + Number +  
 '}';  
 }  
  
 public Admin(String ID, String password, String name, String number) {  
 this.ID = new SimpleStringProperty(ID);  
 this.name = new SimpleStringProperty(name);  
 this.Number = new SimpleStringProperty(number);  
 this.Password = new SimpleStringProperty(password);  
 }}

Class Train manager

package com.example.oopfinalproject;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class TrainManager {  
  
 private ObservableList<Train> list3 = FXCollections.*observableArrayList*();  
  
 public ObservableList<Train> readfromfile(File file) {  
 list3.clear();  
 try (BufferedReader bf = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = bf.readLine()) != null) {  
  
 String[] parts = line.trim().split("\\s+");  
 if (parts.length == 9) {  
 try {  
 int part6 = Integer.*parseInt*(parts[5]);  
 int part7 = Integer.*parseInt*(parts[6]);  
 list3.add(new Train(parts[0], parts[1], parts[2], parts[3], parts[4], part6, part7, parts[7], parts[8]));  
 } catch (NumberFormatException e) {  
 System.*err*.println("Error: Invalid number format in line: " + line);  
 }  
 } else {  
 System.*err*.println("Skipping line due to insufficient data: " + line);  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return list3;  
 }  
  
  
 public ObservableList<Train> getAllBuses() {  
  
 return list3;  
 }  
  
 public static void updateFromField(String busId, String newFrom, File file) {  
 boolean busFound = false;  
 List<String> updatedLines = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
 if (parts.length == 9 && parts[0].equals(busId)) {  
 busFound = true;  
 parts[1] = newFrom;  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 System.*out*.println("Successfully updated 'From' field for bus with ID " + busId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
  
 public static void updateToField(String busId, String newFrom, File file) {  
 boolean busFound = false;  
 List<String> updatedLines = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
  
 if (parts.length == 9 && parts[0].equals(busId)) {  
 busFound = true;  
 parts[2] = newFrom;  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 System.*out*.println("Successfully updated 'From' field for bus with ID " + busId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
  
 public static void updateDateField(String busId, String newDate, File file) {  
 boolean busFound = false;  
 List<String> updatedLines = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
  
 if (parts.length ==9 && parts[0].equals(busId)) {  
 busFound = true;  
 parts[3] = newDate;  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 System.*out*.println("Successfully updated 'From' field for bus with ID " + busId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
 //available seats updation  
 public static void updateAvailseats(String busId, String newDate, File file) {  
 boolean busFound = false;  
 List<String> updatedLines = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
  
 if (parts.length == 9 && parts[0].equals(busId)) {  
 busFound = true;  
 parts[6] = newDate;  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 System.*out*.println("Successfully updated 'From' field for bus with ID " + busId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
 public static void Availseats(String trainId, String newseat, File file) {  
 boolean busFound = false;  
 String up;  
 List<String> updatedLines = new ArrayList<>();  
 int newnumber = Integer.*parseInt*(newseat);  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
 if (parts.length == 9 && parts[0].equals(trainId)) {  
 busFound = true;  
  
 int availableSeats = Integer.*parseInt*(parts[6]);  
 if (availableSeats - newnumber >= 0) {  
 availableSeats -= newnumber;  
 parts[6] = String.*valueOf*(availableSeats);  
 } else {  
 System.*out*.println("No more seats can be updated");  
 return;  
 }  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + trainId + " not found in the list.");  
 }  
 }  
 public static void Cancelseats(String busId, String newseat, File file) {  
 boolean busFound = false;  
 String up;  
 List<String> updatedLines = new ArrayList<>();  
 int newnumber = Integer.*parseInt*(newseat);  
  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
 if (parts.length == 9 && parts[0].equals(busId)) {  
 busFound = true;  
  
 int availableSeats = Integer.*parseInt*(parts[6]);  
 if (availableSeats + newnumber >= 0) {  
 availableSeats += newnumber;  
 parts[6] = String.*valueOf*(availableSeats);  
 } else {  
 System.*out*.println("No more seats can be updated");  
 return;  
 }  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
  
 //time field updation  
 public static void updateTimefield(String busId, String newTime, File file) {  
 boolean busFound = false;  
 List<String> updatedLines = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
  
 if (parts.length == 9 && parts[0].equals(busId)) {  
 busFound = true;  
 parts[4] = newTime;  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 System.*out*.println("Successfully updated 'From' field for bus with ID " + busId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
 public static void updateCancelField(String busId, String newstring, File file) {  
 boolean busFound = false;  
 List<String> updatedLines = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
  
  
 if (parts.length == 9 && parts[0].equals(busId)) {  
 busFound = true;  
 parts[7] = newstring;  
 line = String.*join*(" ", parts);  
 }  
  
 updatedLines.add(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 if (busFound) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (String updatedLine : updatedLines) {  
 writer.write(updatedLine);  
 writer.newLine();  
 }  
 System.*out*.println("Successfully updated 'From' field for bus with ID " + busId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } else {  
 System.*out*.println("Bus with ID " + busId + " not found in the list.");  
 }  
 }  
  
  
 public void updateBus(String busId) {  
 boolean busFound = false;  
  
 for (Train train : list3) {  
 if (train.getBusId().equals(busId)) {  
 busFound = true;  
 System.*out*.println("Bus found! Current details: " );  
 boolean istrue = true;  
 while(istrue) {  
  
 }  
 if(!busFound) {  
 System.*out*.println("Bus with ID " + busId + " not found.");}  
 }}  
 }  
  
}

Class History

package com.example.oopfinalproject;  
  
import javafx.beans.property.SimpleStringProperty;  
  
public class History{  
 private SimpleStringProperty Username;  
 private SimpleStringProperty trainid;  
 private SimpleStringProperty from;  
 private SimpleStringProperty to;  
 private SimpleStringProperty date;  
 private SimpleStringProperty time;  
 private int seats;  
 private SimpleStringProperty fair;  
  
  
 public String getFair() {  
 return fair.get();  
 }  
  
 public SimpleStringProperty fairProperty() {  
 return fair;  
 }  
  
 public void setFair(String fair) {  
 this.fair.set(fair);  
 }  
  
 public String getUsername() {  
 return Username.get();  
 }  
  
 public SimpleStringProperty usernameProperty() {  
 return Username;  
 }  
  
 public void setUsername(String username) {  
 this.Username.set(username);  
 }  
  
 public String getTrainid() {  
 return trainid.get();  
 }  
  
 public SimpleStringProperty TrainidProperty() {  
 return trainid;  
 }  
  
 public void setBusid(String trainid) {  
 this.trainid.set(trainid);  
 }  
  
 public String getFrom() {  
 return from.get();  
 }  
  
 public SimpleStringProperty fromProperty() {  
 return from;  
 }  
  
 public void setFrom(String from) {  
 this.from.set(from);  
 }  
  
 public String getTo() {  
 return to.get();  
 }  
  
 public SimpleStringProperty toProperty() {  
 return to;  
 }  
  
 public void setTo(String to) {  
 this.to.set(to);  
 }  
  
 public String getDate() {  
 return date.get();  
 }  
  
 public SimpleStringProperty dateProperty() {  
 return date;  
 }  
  
 public void setDate(String date) {  
 this.date.set(date);  
 }  
  
 public String getTime() {  
 return time.get();  
 }  
  
 public SimpleStringProperty timeProperty() {  
 return time;  
 }  
  
 public void setTime(String time) {  
 this.time.set(time);  
 }  
  
 public int getSeats() {  
 return seats;  
 }  
  
 public void setSeats(int seats) {  
 this.seats = seats;  
 }  
  
  
 @Override  
 public String toString() {  
 return "History{" +  
 "Username=" + Username +  
 ", trainid=" + trainid +  
 ", from=" + from +  
 ", to=" + to +  
 ", date=" + date +  
 ", time=" + time +  
 ", seats=" + seats +  
 ", fair=" + fair +  
 '}';  
 }  
  
  
  
 public History(String username, String trainid, String from, String to,  
 String date, String time, int seats, String fair) {  
 this.Username =new SimpleStringProperty(username);  
 this.trainid = new SimpleStringProperty(trainid);  
 this.from =new SimpleStringProperty( from);  
 this.to = new SimpleStringProperty(to);  
 this.date =new SimpleStringProperty(date);  
 this.time = new SimpleStringProperty(time);  
 this.seats = seats;  
 this.fair = new SimpleStringProperty(fair);  
 }  
  
}

Class Person

package com.example.oopfinalproject;  
  
import javafx.beans.property.SimpleStringProperty;  
  
public class Person {  
 private SimpleStringProperty ID;  
 private SimpleStringProperty Name;  
 private SimpleStringProperty Number;  
 private SimpleStringProperty Password;  
  
 public String getName() {  
 return Name.get();  
 }  
  
 public SimpleStringProperty nameProperty() {  
 return Name;  
 }  
  
 public void setName(String name) {  
 this.Name.set(name);  
 }  
  
 public String getNumber() {  
 return Number.get();  
 }  
  
 public SimpleStringProperty numberProperty() {  
 return Number;  
 }  
  
 public void setNumber(String number) {  
 this.Number.set(number);  
 }  
  
 public String getID() {  
 return ID.get();  
 }  
  
 public void setID(String ID) {  
 this.ID.set(ID);  
 }  
  
 public String getPassword() {  
 return Password.get();  
 }  
  
 public void setPassword(String password) {  
 this.Password.set(password);  
 }  
  
 @Override  
 public String toString() {  
 return "Person{" +  
 "ID=" + ID +  
 ", Name=" + Name +  
 ", Number=" + Number +  
 ", Password=" + Password +  
 '}';  
 }  
  
 public Person(String ID, String password, String name, String number) {  
 this.ID = new SimpleStringProperty(ID);  
 this.Name = new SimpleStringProperty(name);  
 this.Number = new SimpleStringProperty(number);  
 this.Password = new SimpleStringProperty(password);  
 }}