```
//train station
// w1761265  SE2019281  Mohammed Nazhim Kalam
package sample;
import com.mongodb.BasicDBObject;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.TilePane;
import javafx.stage.Stage;
import org.bson.Document;
import java.io.*;
import java.time.LocalDate;
import java.util.*;

public class TrainStation extends Application {
   public static final int SEATING_CAPACITY = 42;                    //constant representing the maximum
number of seats available in the train
   Stage stage;                                                      //variable stage to display the GUI
   public static int waitingArrayIndex = 0;                          //used for index in waitingRoom
array
```

```
    boolean processDelayAlreadyCreated = false;                                    //used stop running process delay creation several times

    int sizeOfTheWaitingRoom,newStartingPosition,numberOfPassengersInWaitingRoomCurrently; //variables for the waitingRoom implementation

    LocalDate dateSelected;                                                        //value selected from the datePicker

    List<String> waitingRoomPeopleToBeDisplayed = new ArrayList<>();          //this is used to display the passengers in the waiting room after checking in.

    boolean dateCheck,radioCheck = false;                                          //to check if date and radioButton is selected or available

    List<String> loadedTrainQueueData = new ArrayList<>();                    //used to collect loaded data from the mongodb for trainQueue

    String[] selectedRadioButton = new String[1];                              //used to store the selected radioButton value

    List<String> collectLoadedData = new ArrayList<>();                        //used to collect the loaded data

    boolean askDate = true;                                                        //this is used to stop asking the date from the user when ever he enter A again

    List<Passenger> removedPassengerFromQueue = new ArrayList<>();            //this list is used to collect the removed passengers from the list

    List<String> collectDataOnSpecificDate = new ArrayList<>();               //collects the data on the specific date from the loaded data

    String date;                                                                   //gets the user selected date

    Passenger[] waitingRoom = new Passenger[SEATING_CAPACITY];                 //waitingRoom array with passengers inside the array

    PassengerQueue passengerQueue = new PassengerQueue();                      //variable of passengerQueue class to access its content

    List<String> dataToBeSavedIntoTheDatabaseCurrently = new ArrayList<>();    //list to collect the data which has to be saved at the end of the program


    @Override

    public void start(Stage primaryStage){

        stage = primaryStage;                                                  //assigns the stage with the primary stage

        System.out.println("\n----------------------------------------------A S S U M P T I O N S----------------------------------------------");
```

```
        System.out.println("\t* For the report, I have given details of the passengers only who left the waiting room,\n" +
                "\t  which means I have given details related to the passenger who are currently in the train queue\n" +
                "\t  and for the passengers who are boarded into the train. I also have implemented save and load for any date.\n" +
                "\t* I have assumed that every passenger has a unique name.");
        System.out.println("-------------------------------------------------------------------------------------------------------\n");
        displayMenu();                                  //calls the display menu method
    }
    public void displayMenu() {
        System.out.println("    ********************************** W E L C O M E **************************************");        //displays message
        System.out.println("************************ T R A I N  B O A R D I N G  S Y S T E M ****************************\n");    //displays message
        System.out.println("   ************************************** M E N U ***************************************\n");       //displays message
        System.out.println("-->\t\tEnter \"A\" to add a passenger to the trainQueue ");                                          //displays message
        System.out.println("-->\t\tEnter \"V\" to view the trainQueue");                                                        //displays message
        System.out.println("-->\t\tEnter \"D\" to delete passenger from the trainQueue");                                      //displays message
        System.out.println("-->\t\tEnter \"S\" to store trainQueue data into a file");                                        //displays message
        System.out.println("-->\t\tEnter \"L\" to load data back from the file into the trainQueue");                         //displays message
        System.out.println("-->\t\tEnter \"C\" to clear the database of the trainQueue");                                    //displays message
        System.out.println("-->\t\tEnter \"R\" to produce report ");                                                          //displays message
        System.out.println("-->\t\tEnter \"Q\" to exit the program");                                                         //displays message
```

```
    Scanner input = new Scanner(System.in);                    //creating a new scanner variable
    System.out.print("\n>>> \tEnter a option : ");              //displays message
    String enteredOption = input.nextLine();                   //gets the users input and saves it ot
the variable enteredOption
    switch (enteredOption.toLowerCase()) {
      case ("a"):                                              //checks if the user has entered "a" or "A" and
proceed
        addingPassengerToTrainQueue();                         //calls the
addingPassengerToTrainQueue method
        break;
      case ("v"):                                              //checks if the user has entered "v" or "V" and
proceed
        viewTrainQueue();                                      //calls the viewTrainQueue method
        break;
      case ("d"):                                              //checks if the user has entered "d" or "D" and
proceed
        deletePassengerFromTrainQueue();                       //calls the
deletePassengerFromTrainQueue method
        break;
      case ("s"):                                              //checks if the user has entered "s" or "S" and
proceed
        storeTrainQueueData();                                 //calls the storeTrainQueueData method
        break;
      case ("l"):                                              //checks if the user has entered "l" or "L" and
proceed
        loadTrainQueueData();                                  //calls the loadTrainQueueData method
        break;
      case ("c"):                                              //checks if the user has entered "c" or "C" and
proceed
        clearDataBase();                                       //calls the clearDataBase method
        break;
```

```
        case ("r"):                                     //checks if the user has entered "r" or "R" and
proceed
            displayReport();                            //calls the displayReport method
            break;
        case ("q"):                                     //checks if the user has entered "q" or "Q" and
proceed
            System.out.println("Thanks for visiting Denuwara Express. \nProgram exiting...");
            System.exit(1);                             //exits the program
        default:
            System.out.println("You have entered an incorrect option...\nPlease enter a correct option
from (A,V,D,S,L,C,R,Q) only...\n\n");  //displays message                  //displays message
            displayMenu();                              //calls the displayMenu method
    }
  }
  public void datePicker() {                            //used to select a date from the date picker in
order to load the necessary details
    DatePicker datePicker = new DatePicker();                   //creates the datePicker
    datePicker.setOnMouseDragEntered(event -> datePicker.setCursor(Cursor.HAND));
    datePicker.getEditor().setDisable(true);                    //used to disable past dates
    datePicker.setDayCellFactory(picker -> new DateCell() {   //this part disables the old dates in the
datepicker
        public void updateItem(LocalDate date, boolean empty) {         //used to update the
datePicker with the current date
            super.updateItem(date, empty);
            LocalDate today = LocalDate.now();                  //gets today's date
            setDisable(empty || date.compareTo(today) < 0 );
        }
    });
    datePicker.setLayoutX(140);                             //setting layout for the datePicker
    datePicker.setLayoutY(130);                             //setting layout for the datePicker
```

```
    datePicker.setStyle("-fx-background-color:brown;");                    //setting style for the
datePicker


    Label dateLabel = new Label("SELECT DATE AND JOURNEY");              // creating a date label
    dateLabel.setLayoutX(70);                                          // setting layout for the dateLabel
    dateLabel.setLayoutY(40);                                          // setting layout for the dateLabel
    dateLabel.setAlignment(Pos.CENTER);
    dateLabel.setMinSize(320,40);                        // setting size for the dateLabel
    dateLabel.setStyle("-fx-font-size:18; -fx-border-color:brown; -fx-font-weight:bold; -fx-border-
radius:20; -fx-border-width:2;"); // setting style for the dateLabel


    Button okButton = new Button("OK");                                 //creating OK button for the GUI
    okButton.setMinWidth(70);                                          // setting minimum width for the button
    okButton.setLayoutX(360);                                          // setting layout for the button
    okButton.setLayoutY(280);                                          // setting layout for the button
    okButton.setStyle("-fx-background-radius:20; -fx-border-color:black; -fx-border-radius:20;");    //
setting style for the button


    ToggleGroup toggleGroup = new ToggleGroup();                        //used to create a toggle
group for the radio buttons
    RadioButton radioButtonForReturn = new RadioButton("Return journey");      //creating radio
button for return journey
    radioButtonForReturn.setToggleGroup(toggleGroup);                    //adding the created radio
button into the toggle group
    radioButtonForReturn.setLayoutY(230);                              // setting layout for the
radioButton
    radioButtonForReturn.setLayoutX(270);                              // setting layout for the
radioButton


    RadioButton radioButtonForNormal = new RadioButton("Not return journey");   //creating radio
button for not return journey
```

```
    radioButtonForNormal.setToggleGroup(toggleGroup);                    //adding the created radio button into the toggle group

    radioButtonForNormal.setLayoutX(50);                                 // setting layout for the radioButton

    radioButtonForNormal.setLayoutY(230);                                // setting layout for the radioButton


    radioButtonForNormal.setOnAction(event -> {                          //setting action for the radio button

        selectedRadioButton[0] = radioButtonForNormal.getText();         //getting the radio button value to an array

    });

    radioButtonForReturn.setOnAction(event -> {                          //setting action for the radio button

        selectedRadioButton[0] = radioButtonForReturn.getText();         //getting the radio button value to an array

    });

    okButton.setOnAction(event -> {                                      //this is the ok button where the main program runs

        Alert informationBox = new Alert(Alert.AlertType.INFORMATION);   //creating a information box

        dateSelected = datePicker.getValue();                            //gets the date selected by the user on the datePicker

        if (dateSelected == null && selectedRadioButton[0] == null) {    //checks if the date and radio button values are null

            informationBox.setHeaderText("Please select a radio button and select a date from the date picker");    //texts inside the information box

            informationBox.setTitle("Error");                            //setting a title to the information box

            informationBox.showAndWait();                                //displays the alert box

        } else {

            if (dateSelected == null) {
```

```
            informationBox.setHeaderText("Please select a date from the date picker");          //texts inside the information box
            informationBox.setTitle("Error");                                                   //setting a title to the information box
            informationBox.showAndWait();                                                       //displays the alert box
        } else {
            dateCheck = true;                                                                   //sets dateCheck to true
        }
        if (selectedRadioButton[0] == null) {                                                   //checks if the radio button is not selected
            informationBox.setHeaderText("Please select a radio button");                       //texts inside the information box
            informationBox.setTitle("Error");                                                   //setting a title to the information box
            informationBox.showAndWait();                                                       //displays the alert box
        } else {
            radioCheck = true;                                                                  //sets the radioCheck to true
        }
    }
    if (dateCheck && radioCheck) {                                                              //checks if the dataCheck and radioCheck is true
        askDate = false;                                                                        //assigns askDate to false so no need to ask date again
        boolean dateFoundInLoadSection = false;                                                 //check if the loaded data has the date selected
        date = String.valueOf(dateSelected);                                                    //stores the selected date into a variable
        for(String loadData : loadedTrainQueueData){                                            //looping through the loaded data one by one

            String[] splitOnADate = loadData.split(",");                                        //separating the data on a specific date
```

```
            String[] splitDataOnADate = splitOnADate[1].split("=");         //extracts the relevant data from the 1st position

            String[] splitJourneyDataOnADate = splitOnADate[2].split("=");    //extracts the relevant data from the 2nd position


            if(date.equals(splitDataOnADate[1].trim()) && splitJourneyDataOnADate[1].trim().equals(selectedRadioButton[0])){  //checks if date and radio button is equal to the selected value by the user

                dateFoundInLoadSection = true;                                  //assigns dateFoundInLoadSection to true since data found in load

                date = splitDataOnADate[1].trim();                              //date inserting into the variable

                selectedRadioButton[0] = splitJourneyDataOnADate[1].trim();             //radio insertion into the variable


                String[] newStartingPositionArray = splitOnADate[3].split("="); //newStartingDateOnADate splitting

                newStartingPosition = Integer.parseInt(newStartingPositionArray[1].trim()); //newStartingDateOnDate insertion into the variable


                String[] waitingArrayIndexArray = splitOnADate[4].split("");        //waitingArrayIndex splitting

                waitingArrayIndex = Integer.parseInt(waitingArrayIndexArray[1].trim()); //waitingArrayIndex insertion into the variable


                String[]  numberOfPassengersInWaitingRoomCurrentlyArray = splitOnADate[5].split("="); //numberOfPassengersInWaitingRoomCurrently splitting

                numberOfPassengersInWaitingRoomCurrently = Integer.parseInt(numberOfPassengersInWaitingRoomCurrentlyArray[1].trim()); //numberOfPassengersInWaitingRoomCurrently insertion into the variable

                removedPassengerFromQueue.clear();                                  //clears the removedPassengerFromQueue list


                for(int counter = 6; counter<174; counter += 4){                    //loops to collect data for removedPassengerFromQueue
```

```
                String[] removedPassengerFromQueueArrayFirstName =
splitOnADate[counter].split("=");    //splits the data

                if(!removedPassengerFromQueueArrayFirstName[1].equals("null")){
//checks if the first data is not null

                    Passenger passenger = new Passenger();                                    //creating
passenger object

                    passenger.setFirstName(removedPassengerFromQueueArrayFirstName[1].trim());
//setting loaded first name


                    String[] removedPassengerFromQueueArrayLastName =
splitOnADate[counter+1].split("=");        //splits the data

                    passenger.setSurname(removedPassengerFromQueueArrayLastName[1].trim());
//setting loaded surname


                    String[] removedPassengerFromQueueArraySeconds =
splitOnADate[counter+2].split("=");         //splits the data


passenger.setSecondsInQueue(Integer.parseInt(removedPassengerFromQueueArraySeconds[1].trim()));
//setting loaded waitingTime


                    String[] removedPassengerFromQueueArraySeatNumber =
splitOnADate[counter+3].split("=");      //splits the data


passenger.setSeatNumber(Integer.parseInt(removedPassengerFromQueueArraySeatNumber[1].trim()));
//setting loaded seat number

                    removedPassengerFromQueue.add(passenger);
//adds the passenger into the list

                } else{

                    break;

                }

            }

            int waitingRoomInsertionIndex = 0;                                    //index for the
looping below

            Arrays.fill(waitingRoom, null);
```

```
                for(int counter = 174; counter<342; counter+=4){                    //loop used to add loaded passengers to the waiting room
                    String[] waitingRoomArrayFirstName = splitOnADate[counter].split("=");          //splits the data
                    if(!waitingRoomArrayFirstName[1].equals("null")){                    //checks if the first data is not null
                        Passenger passenger = new Passenger();                    //creates passenger object
                        passenger.setFirstName(waitingRoomArrayFirstName[1].trim());
//setting loaded firstName

                        String[] waitingRoomArrayLastName = splitOnADate[counter+1].split("=");       //splits the data
                        passenger.setSurname(waitingRoomArrayLastName[1].trim());
//setting loaded surname

                        String[] waitingRoomArraySeconds = splitOnADate[counter+2].split("=");        //splits the data
                        passenger.setSecondsInQueue(Integer.parseInt(waitingRoomArraySeconds[1].trim()));
//setting loaded waiting time

                        String[] waitingRoomArraySeatNumber = splitOnADate[counter+3].split("=");
//splits the data
                        passenger.setSeatNumber(Integer.parseInt(waitingRoomArraySeatNumber[1].trim()));
//setting loaded seatNumber

                        waitingRoom[waitingRoomInsertionIndex] = passenger;                    //adds the passenger into the waiting room array
                        sizeOfTheWaitingRoom = waitingRoomInsertionIndex + 1;
//increments the sizeOfTheWaitingRoom variable
                    } else{
                        break;
                    }
```

```
                waitingRoomInsertionIndex++;

            }

            String[] PassengerQueueSetFirst = splitOnADate[342].split("=");                    //splits the
data for passengerQueue variables

            passengerQueue.setFirst(Integer.parseInt(PassengerQueueSetFirst[1].trim()));
//setting loaded first value


            String[] PassengerQueueSetLast = splitOnADate[343].split("=");                     //splits the
data for passengerQueue variables

            passengerQueue.setLast(Integer.parseInt(PassengerQueueSetLast[1].trim()));
//setting loaded last value


            String[] PassengerQueueSetMaxLength = splitOnADate[344].split("=");                //splits
the data for passengerQueue variables


passengerQueue.setMaxLength(Integer.parseInt(PassengerQueueSetMaxLength[1].trim()));
//setting loaded max length


            String[] PassengerQueueSetSize = splitOnADate[345].split("=");                     //splits the
data for passengerQueue variables

            passengerQueue.setSize(Integer.parseInt(PassengerQueueSetSize[1].trim()));
//setting loaded size value

            passengerQueue.updatedPassengersInQueue.clear();                                   //clears
the updatedPassengersInQueue list


            for(int index = 346; index<388; index++){                                          //loop used to add
loaded passengers to the updatedPassengersInQueue list

                String[] updatedPassengerInQueueArray = splitOnADate[index].split("=");
//splitting data

                if(!updatedPassengerInQueueArray[1].equals("null")) {                          //checks if
data is not equal to null


passengerQueue.updatedPassengersInQueue.add(Integer.valueOf(updatedPassengerInQueueArray[1].tr
im()));  //setting the data
```

```
                } else{
                    break;
                }
            }
            int position = 0;
            for(int index = 0; index<SEATING_CAPACITY; index++){                    //used to
set null for all passengers in the train queue
                passengerQueue.setQueueArray(null,index);
            }
            for(int index = 388; index<556; index+=4){                              //loop used to add
loaded passengers to the queueArray in the passenger queue class
                String[] queueArrayFirstName = splitOnADate[index].split("=");      //splits the
data
                if(!queueArrayFirstName[1].equals("null")){                         //checks if the
first data is not null
                    Passenger passenger = new Passenger();                          //creates the
passenger object
                    passenger.setFirstName(queueArrayFirstName[1].trim());          //setting
loaded firstName

                    String[] queueArrayLastName = splitOnADate[index+1].split("="); //splits the
data
                    passenger.setSurname(queueArrayLastName[1].trim());             //setting
loaded surName

                    String[] queueArraySeconds = splitOnADate[index+2].split("=");  //splits the
data
                    passenger.setSecondsInQueue(Integer.parseInt(queueArraySeconds[1].trim()));
//setting loaded waiting time

                    String[] queueArraySeatNumber = splitOnADate[index+3].split("="); //splits
the data
```

```
                    passenger.setSeatNumber(Integer.parseInt(queueArraySeatNumber[1].trim())); //setting loaded seatNumber


                    passengerQueue.setQueueArray(passenger,position);                    //sets the data

                    passengerQueue.setArrayPositionIndex(position);                    //sets the arrayPositionIndex

                    position++;                                                    //incrementation

                  } else{

                    break;

                  }

                }

                String[] processDelayAlreadyCreatedArray = splitOnADate[556].split("="); //processDelay data splitting

                processDelayAlreadyCreated = Boolean.parseBoolean(processDelayAlreadyCreatedArray[1].replace("}}","").trim());  //processDelay data  setting

              }

            }

            if(dateFoundInLoadSection){                                    //checks if data found in the loaded data

              stage.close();                                              //closes the stage and calls the below method

              trainQueue();

            } else {

              MongoClient mongoClient = new MongoClient("localhost", 27017);          //this is used to load the CW_01 details

              if (selectedRadioButton[0].equals("Not return journey")) {              //checks radio button value

                MongoDatabase database = mongoClient.getDatabase("CourseWorkDB"); //creates a new database
```

```
            MongoCollection<Document> CustomerNameCollection =
database.getCollection("CustomerNameCollection");    //creates a new collection for the
seatBookingArray

            collectLoadedData.clear();

            for (Document dataSet : CustomerNameCollection.find()) {

               collectLoadedData.add(String.valueOf(dataSet));          //gets all the records stored from
the collection and puts it into this list

            }

            boolean containsData = false;

            for (String records : collectLoadedData) {                          //used to access all the records
in this list

               String[] split = records.split(",");                     //used to split the data by ","

               split[1] = split[1].substring(6, 16);                          //date extracted

               split[43] = split[43].replace("}}", "");          //used to remove }}

               if (date.equals(split[1])) {                                 //checks with the selected date

                  for (int index = 2; index < split.length; index++) {

                     String[] innerSplit = split[index].split("=");         //splitting the data

                     if (!innerSplit[1].equals(" ")) {                     //this part is used to check if bookings
are available on this date

                        containsData = true;                               //if it contains then sets containsData
to true

                        break;

                     }

                  }

                  if (containsData) {                                       //if variable true

                     for (int index = 2; index < split.length; index++) {

                        collectDataOnSpecificDate.add(split[index]);              //collect the loaded data
into a variable

                     }

                     stage.close();                                        //this thing closes the stage when OK is
clicked

                  }
```

```
                }
            }
            if (!containsData) {                                    //if no data on the selected date
                informationBox.setHeaderText("Please select a different date because there are no
bookings made \n" +
                        "on this date");                            //texts inside the information box
                informationBox.setTitle("Error");                   //setting a title to the
information box
                informationBox.showAndWait();                       //displays the alert box
                dateCheck = false;
            } else {
                passengersChecking();                               //calls the passengersChecking
method
            }
        } else {       //if return journey radio button is selected
            MongoDatabase databaseReturn = mongoClient.getDatabase("CourseWorkDBReturn");
//creates a new database to get data from a database
            MongoCollection<Document> CustomerNameCollectionForReturn =
databaseReturn.getCollection("CustomerNameCollectionForReturn");  //creates a new collection to get
data from a collection
            collectLoadedData.clear();                              //clears the collectLoadedData list
            for (Document dataSet : CustomerNameCollectionForReturn.find()) {
                collectLoadedData.add(String.valueOf(dataSet));     //gets all the records stored from
the collection and puts it into this list
            }
            date = String.valueOf(dateSelected);                    //stores the selected date into a
variable
            boolean containsData = false;
            for (String records : collectLoadedData) {              //used to access all the records in this
list
                String[] split = records.split(",");                //used to split the data by ","
                split[1] = split[1].substring(6, 16);               //date extracted
```

```
                split[43] = split[43].replace("}}", "");            //used to remove }}
                if (date.equals(split[1])) {
                    for (int index = 2; index < split.length; index++) {              //loops through the data on
that date if equal to selected date
                        String[] innerSplit = split[index].split("=");          //splitting data
                        if (!innerSplit[1].equals(" ")) {                        //checks if there is data or not
                            containsData = true;
                            break;
                        }
                    }
                    if (containsData) {
                        for (int index = 2; index < split.length; index++) {           //if data is present
                            collectDataOnSpecificDate.add(split[index]);              //collect the loaded data
into a variable
                        }
                        stage.close();                                    //this thing closes the stage when OK is
clicked
                    }
                }
            }
            if (!containsData) {
                informationBox.setHeaderText("Please select a different date because there are no
bookings made \n" +
                        "on this date");                          //texts inside the information box
                informationBox.setTitle("Error");                    //setting a title to the information box
                informationBox.showAndWait();                         //displays the alert box
                dateCheck = false;
            } else {
                passengersChecking();                              //calls the passengersChecking method
            }
```

```
            }
          }
        }
    });
    AnchorPane anchorPane = new AnchorPane();                                   //creating an anchorPane
    anchorPane.setStyle("-fx-background-color:#FFD3B5;");                       //setting style for the anchorPane
    anchorPane.getChildren().addAll(datePicker, dateLabel, radioButtonForNormal, radioButtonForReturn, okButton);   //adding nodes to this pane
    Scene scene = new Scene(anchorPane, 450, 350);                              //creating the scene
    stage.setTitle("Run Simulation");                                           //setting title to stage
    stage.setScene(scene);                                                      //setting the scene to the stage
    stage.show();                                                               //makes the stage visible
    stage.setOnCloseRequest(event -> {                                          //setting on close request
      event.consume();                          //from this point the closeProgram method will handle the controls
      closeProgram();                           //calls the closeProgram function
    });
  }
  public void passengersChecking(){             //in this method I will add all the passengers who check in only into the waiting room
    System.out.println("--------------------CHECKING INTO WAITING ROOM PROCESS--------------------\n");
    System.out.println("If the passenger has checked in please enter \"Y\" or else enter \"N\". \n");
    for(String index:collectDataOnSpecificDate){                //looping through the records of that specific date
      String[] seatNumberAndName = index.split("=");            //splitting data
      if(!seatNumberAndName[1].equals(" ")){                    //checking if there is a customer booked in this particular date
```

```
        String checkingInfo = "";                                   //used to get the user input for check in to the waiting room.

        while (!checkingInfo.toLowerCase().equals("y") && !checkingInfo.toLowerCase().equals("n")) { //loops if user didn't enter "y" or "n"

            System.out.print("Did passenger " + seatNumberAndName[1] + " check in for seat number " + seatNumberAndName[0].trim() + " = ");

            Scanner input = new Scanner(System.in);                 //creates a scanner

            checkingInfo = input.nextLine();                        //gets the users input

            if (!checkingInfo.toLowerCase().equals("y") && !checkingInfo.toLowerCase().equals("n")) { //checks if user didn't enter "y" or "n"

                System.out.println("Please enter only \"Y\" or \"N\" only. ");                   //display message

            }

            if(checkingInfo.toLowerCase().equals("y")){                 //if user has entered "y"

                waitingRoomPeopleToBeDisplayed.add(seatNumberAndName[1]);           //names of the passengers who checked in and added to this list

                String[] totalNames = seatNumberAndName[1].split(" ");      //splitting the names from the space inorder to get the other names

                String firstName = totalNames[0];                       //first Name of the passenger

                StringBuilder surname = new StringBuilder();                //surname of the passenger

                for(int count = 1; count<(totalNames.length); count++){

                    surname.append(totalNames[count]).append(" ");              //we append the remaining names as the surname

                }

                Passenger passenger = new Passenger();                          //creating the passenger object

                passenger.setFirstName(firstName);                              //setting firstName to passenger

                passenger.setSurname(String.valueOf(surname));                          //setting surname to passenger

                passenger.setSeatNumber(Integer.parseInt(seatNumberAndName[0].trim())); //setting seatNumber
```

```
                waitingRoom[waitingArrayIndex] = passenger;                    //adding the passenger to the waitingRoom
                waitingArrayIndex++;                                           //incrementing
            }
        }
    }
    System.out.println("\nThese are the list of passengers who have checked in and are sent to the waiting room: \n");
    for (String customer : waitingRoomPeopleToBeDisplayed){          //loops and displays the passengers who have only check into the waiting room
        System.out.println(" * " + customer);
    }
    try {
        Thread.sleep(3000);                          //time delay to display the scene
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    numberOfPassengersInWaitingRoomCurrently = waitingArrayIndex;          //set the numberOfPassengers in the waiting to a variable
    sizeOfTheWaitingRoom = waitingArrayIndex;                    //set the numberOfPassengers in the waiting to a variable
    trainQueue();                                    //calls the trainQueue method
  }
  public void trainQueue(){
    if(numberOfPassengersInWaitingRoomCurrently==0){                    //checks if the waiting room is empty
        Alert informationBox = new Alert(Alert.AlertType.INFORMATION);          //creating a information box
        informationBox.setHeaderText("The waiting room is empty, therefore there are no passengers to be moved \ninto the train queue"); //texts inside the information box
        informationBox.setTitle("Error");                    //setting a title to the information box
```

```
        informationBox.showAndWait();                                   //displays the alert box

    } else {

        Random random = new Random();                                   //creates the random variable

        int transferNumber = random.nextInt(6) + 1;                     //generating the transfer
number

        if (transferNumber > numberOfPassengersInWaitingRoomCurrently) {       //checks if the transfer
number is greater than the number of passengers in the waiting room

            Alert informationBox = new Alert(Alert.AlertType.INFORMATION);     //creating a information
box

            informationBox.setHeaderText(transferNumber + " number of passengers were planned to be
moved to the train queue but \n" +

                    "there are only " + numberOfPassengersInWaitingRoomCurrently + " number of
passengers in the waiting room therefore we are moving\n"

                    + numberOfPassengersInWaitingRoomCurrently + " passengers from the waiting room to
the train queue");     //texts inside the information box

            informationBox.setTitle("Details");                         //setting a title to the
information box

            informationBox.showAndWait();                               //displays the alert box

            transferNumber = numberOfPassengersInWaitingRoomCurrently;          //sets the
number of passengers in waiting room to transfer number

            numberOfPassengersInWaitingRoomCurrently = 0;               //makes the waiting
room empty

        } else {

            numberOfPassengersInWaitingRoomCurrently -= transferNumber;         //reduces
the number of passengers in the waiting room by the transfer number

        }

        for (int index = newStartingPosition; index < (transferNumber + newStartingPosition); index++) {
//loops and moves that number of passengers from waiting room to the passenger queue

            passengerQueue.add(waitingRoom[index]);             //adding passenger from waiting room
to the queue

        }

        newStartingPosition += transferNumber;                  //updates the new starting position
```

```
        passengerQueue.setUpdatedPassengersInQueue();              //updates the passengers in the train queue
    }
    trainQueueVisualization();                                     //calls this method
    if(passengerQueue.getUpdatedPassengersInQueue().size() == SEATING_CAPACITY){          //checks if train queue is full (42)
        Alert informationBox = new Alert(Alert.AlertType.INFORMATION);                 //creating a information box
        informationBox.setHeaderText("Now the train queue is full");                   //texts inside the information box
        informationBox.setTitle("Error");                                              //setting a title to the information box
        informationBox.showAndWait();                                                  //displays the alert box
    }
  }
  public void trainQueueVisualization() {
    Label headingQueue = new Label("ADDING PASSENGER TO QUEUE");
    headingQueue.setLayoutX(100);                                  //setting position
    headingQueue.setLayoutY(40);                                   //setting position
    headingQueue.setMinWidth(550);
    headingQueue.setAlignment(Pos.CENTER);                                 //position alignment
    headingQueue.setStyle(" -fx-text-fill:DARKBLUE; -fx-font-size:30px; -fx-font-style:italic; -fx-background-radius:20; -fx-border-radius:20; " +
            "-fx-border-color:red; -fx-border-width:5;");


    Label note = new Label("NOTE:  The red colour circle represents a passenger.\n" +          //label with necessary details
            "          The number on the red circle represents the seat number of the passenger.");
    note.setLayoutX(50);                                                   //setting layout position
    note.setLayoutY(150);                                                  //setting layout position
    note.setStyle("-fx-font-size:18px; -fx-font-weight:bold; -fx-font-style:italic;");          //setting style
```

```
    TilePane queueFormation = new TilePane();                                   //tilePane for the queue formation
    queueFormation.setMinSize(1250, 30);                                        //setting min width and min height
    queueFormation.setLayoutX(60);                                              //setting layout position
    queueFormation.setLayoutY(360);                                             //setting layout position


    Label waitingRoomLabel = new Label("WAITING ROOM");      //--------------------waitingRoomLabel----------
    waitingRoomLabel.setStyle("-fx-background-color:#002366; -fx-text-fill:white; -fx-alignment:center; -fx-font-size:40px; " +
            "-fx-background-radius: 50; -fx-border-color:red; -fx-border-radius:50; -fx-font-style:italic; -fx-border-width:3; ");        //setting style
    waitingRoomLabel.setLayoutX(770);                                           //setting layout position
    waitingRoomLabel.setLayoutY(20);                                            //setting layout position
    waitingRoomLabel.setMinSize(500, 250);


    Label counter = new Label("COUNTER");                                       // counter LABEL
    counter.setMinSize(180, 50);                                        // setting size
    counter.setStyle(" -fx-rotate:270; -fx-background-color:brown; -fx-text-fill:white; -fx-background-radius: 35;" +
            " -fx-alignment:center; -fx-font-size:25px; -fx-border-color:black; -fx-border-radius:50; -fx-border-width:2;");    //setting style
    counter.setLayoutY(350);                                                    //setting layout position
    counter.setLayoutX(-60);                                                    //setting layout position
    if(passengerQueue.getUpdatedPassengersInQueue().size()!=0) {                //checks if the list size is equal to 0
```

```
        for (int indexPassenger : passengerQueue.getUpdatedPassengersInQueue()) {          //creates buttons for the passengers (using the seat number) in the list

            Button customer;                                                    //creates a button

            if (indexPassenger < 10) {                                          //if the value is less than 10, adding "0" with the value

                customer = new Button("0" + String.valueOf(indexPassenger));

            } else {

                customer = new Button(String.valueOf(indexPassenger));

            }

            customer.setStyle("-fx-background-radius:40; -fx-background-color:red; -fx-border-radius:40;"); //setting button style

            queueFormation.getChildren().addAll(customer);                      //adding the buttons into the tilePane

            AnchorPane anchorPane = new AnchorPane();                           //creating an anchorPane

            anchorPane.setStyle("-fx-background-image: url('File:/C:/Users/Nazhim/Desktop/pp2_cw2/src/sample/one.jpg');"); //adding the image

            anchorPane.getChildren().addAll(waitingRoomLabel, counter, queueFormation,note,headingQueue);          //adding nodes into the pane

            Scene scene = new Scene(anchorPane, 1300, 500);                     //creating scene and setting the size

            stage.setScene(scene);                                              // adding the scene into the stage

            stage.setTitle("Adding passenger to queue");                        //setting the stage title

            stage.show();                                                       //displaying the stage

        }

    } else{                                                                     //if the list is empty then i display an empty pane with no train queue

        AnchorPane anchorPane = new AnchorPane();                               //creating an anchor pane

        anchorPane.setStyle("-fx-background-image: url('File:/C:/Users/Nazhim/Desktop/pp2_cw2/src/sample/one.jpg');");  //adding background image

        anchorPane.getChildren().addAll(waitingRoomLabel, counter, queueFormation,note,headingQueue);          //adding nodes into the pane
```

```
        Scene scene = new Scene(anchorPane, 1300, 500);                    //creating a scene and adding the anchor pane
        stage.setScene(scene);                                             //setting the scene to the stage
        stage.setTitle("Adding passenger to queue");                       //setting title to the stage
        stage.show();                                                      //displaying the stage
    }
  }
  public void addingPassengerToTrainQueue() {
    if (askDate){                                //checks if the askDate is equal true, (user already have selected a date)
        datePicker();                            //calling the datePicker method
    } else{
        trainQueue();                            //calling the trainQueue method
    }
  }
  public void viewTrainQueue(){
    Label noteForWaiting = new Label("NOTE: Red colour represents an available passenger\n Grey colour represents an empty seat");
    noteForWaiting.setLayoutY(560);                                        //setting layout position
    noteForWaiting.setLayoutX(30);                                         //setting layout position
    noteForWaiting.setStyle("-fx-font-weight:bold; -fx-font-size:15px; -fx-font-style:italic;"); //setting style

    AnchorPane anchorPaneWaiting = new AnchorPane();                       //creating anchorPane
    GridPane gridPaneWaiting = new GridPane();                             //creating gridPane
    gridPaneWaiting.setMinSize(1000,500);                                  //setting size
    gridPaneWaiting.setLayoutY(50);                                        //setting layout position
```

```
    gridPaneWaiting.setLayoutX(100);                                    //setting layout
position
    gridPaneWaiting.setAlignment(Pos.CENTER);                           //setting
alignment
    gridPaneWaiting.setHgap(50);                                        //setting Horizontal
gap
    gridPaneWaiting.setVgap(15);                                        //setting Vertical gap


    Button buttonToTrainQueue = new Button("click for train queue");    //
creating the button
    buttonToTrainQueue.setLayoutY(560);                                 //setting layout
position
    buttonToTrainQueue.setLayoutX(1000);                                //setting layout
position
    buttonToTrainQueue.setStyle("-fx-background-radius:20; -fx-border-radius:20; -fx-border-
color:black; -fx-border-width:2;"); //setting style


    Label heading = new Label("WAITING ROOM");                          //creating a label
    heading.setStyle(" -fx-text-fill:DARKBLUE; -fx-font-size:40px; -fx-background-radius:20; -fx-border-
radius:20; " + //setting style
        "-fx-border-color:red; -fx-border-width:5; -fx-font-style:italic;");
    heading.setLayoutY(10);                                             //setting layout position
    heading.setMinWidth(400);                                           //setting width
    heading.setAlignment(Pos.CENTER);                                   //setting alignment
    heading.setLayoutX(400);                                            //setting layout position


    Label[] labelForNamesArray = new Label[SEATING_CAPACITY];
//creating a label array for names
    Button[] passengerButtonArray = new Button[SEATING_CAPACITY];
//creating a button array
    int rowLabel = 0;                       //variables for button creation
    int columnLabel = 0;                    //variables for button creation
```

```
int labelNameIndex = 0;                          //variables for button creation
int rowButton = 0;                              //variables for button creation
int columnButton = 1;                            //variables for button creation
int passengerButtonIndex = 0;                          //variables for button creation


for(int rows=0; rows<6; rows++){
   for(int labelCount=0; labelCount<7; labelCount++){                                    //loop for creating labels
      labelForNamesArray[labelNameIndex] = new Label("Empty Seat");
//initialize the label with empty seat
      labelForNamesArray[labelNameIndex].setTranslateY(15);
      gridPaneWaiting.add(labelForNamesArray[labelNameIndex],rowLabel,columnLabel);
//adding the label into the gridPaneWaiting
      labelNameIndex++;
      rowLabel++;
    }
    for(int buttonCount=0; buttonCount<7; buttonCount++){
      passengerButtonArray[passengerButtonIndex] = new Button("X");
//creating buttons with "X" letter for all
      passengerButtonArray[passengerButtonIndex] .setStyle("-fx-background-radius:10; -fx-border-color:black; -fx-border-radius:10;" +
          " -fx-background-color:grey; -fx-text-fill:white;");
      passengerButtonArray[passengerButtonIndex].setMinSize(40,5);              //setting size
      passengerButtonArray[passengerButtonIndex] .setTranslateY(5);
      gridPaneWaiting.add(passengerButtonArray[passengerButtonIndex]
,rowButton,columnButton);          // adding the buttons into the gridPane
      passengerButtonIndex++;
      rowButton++;
    }
    rowButton = 0;        //set of variable for making the layout
    rowLabel = 0;
```

```
            columnButton+=2;

            columnLabel+=2;

        }

        for(int count = newStartingPosition;
count<(newStartingPosition+numberOfPassengersInWaitingRoomCurrently); count++){  //setting
passengers in the waiting room currently

            passengerButtonArray[count].setStyle("-fx-background-color:red; -fx-background-radius:10; -fx-
border-color:black; -fx-border-radius:10; fx-text-fill:white;");

            labelForNamesArray[count].setText(waitingRoom[count].getFirstName() +
waitingRoom[count].getSurname());

        }

        buttonToTrainQueue.setOnAction(event -> {                                    //actions when
buttonToTrainQueue is clicked

            Label note = new Label("NOTE:  The grey colour circle represents an empty slot in the queue\n" +
//creates a label

                    "           Red colour circle represent the passenger in the train queue.");

            note.setLayoutX(50);

            note.setLayoutY(160);

            note.setStyle("-fx-font-size:15px; -fx-font-weight:bold; -fx-font-style:italic;");
//setting styles

            stage.close();

            try {                                                    // displaying the stage after a time
delay

                Thread.sleep(500);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

            AnchorPane anchorPaneQueue = new AnchorPane();                    //creating an
anchorPane and adding the node

            anchorPaneQueue.getChildren().addAll(note);


            Label counterLabel = new Label("COUNTER");                    //creates the counter label
```

```
        counterLabel.setMinSize(180, 50);                    // setting the size to the label
        counterLabel.setStyle(" -fx-rotate:270; -fx-background-color:brown; -fx-text-fill:white; -fx-background-radius: 35;" +
                " -fx-alignment:center; -fx-font-size:25px; -fx-border-color:black; -fx-border-radius:50; -fx-border-width:2;");
        counterLabel.setTranslateY(350);                                    //setting position
        counterLabel.setLayoutX(-64);                                      //setting position

        TilePane tilePane = new TilePane();                                  //creating tilePane
        tilePane.setMinSize(1220,30);                        //setting the size
        tilePane.setLayoutY(365);                                      //setting position
        tilePane.setLayoutX(52);                                      //setting position

        Button buttonToTrainSeats = new Button("click for train seats");          // creating a button
        buttonToTrainSeats.setLayoutY(490);                                    //setting position
        buttonToTrainSeats.setLayoutX(1140);                                    //setting position
        buttonToTrainSeats.setStyle("-fx-background-radius:20; -fx-border-radius:20; -fx-border-color:black; -fx-border-width:2;");

        Label headingQueue = new Label("TRAIN QUEUE");
        headingQueue.setLayoutX(500);                                    //setting position
        headingQueue.setLayoutY(60);                                    //setting position
        headingQueue.setMinWidth(320);
        headingQueue.setAlignment(Pos.CENTER);                                    //position alignment
        headingQueue.setStyle(" -fx-text-fill:DARKBLUE; -fx-font-size:40px; -fx-background-radius:20; -fx-border-radius:20; " +
                "-fx-border-color:red; -fx-border-width:5; -fx-font-style:italic;");

        for(int count=0; count<SEATING_CAPACITY; count++){                    //this loop is used to create the button, set style and add them to the pane
```

```
            passengerButtonArray[count] = new Button("00");
            passengerButtonArray[count] .setStyle("-fx-background-color:grey; -fx-background-radius:40; -fx-border-radius:40; -fx-text-fill:white;");
            tilePane.getChildren().addAll(passengerButtonArray[count] );
        }
        int xPositionOfName = 0;
        String passengerName = "";
        for(int count=0; count<SEATING_CAPACITY; count++){                  //loops to create the label in a proper layout and position it properly
            passengerName = "Empty";
            labelForNamesArray[count] = new Label(passengerName);           //initializes all the label with "Empty"
            int lengthOfName = passengerName.length();
            double varyPosition = 1;
            varyPosition = lengthOfName*4;
            labelForNamesArray[count] .setStyle("-fx-rotate:270; -fx-wrap-text:true;");       //setting the style for the labels
            labelForNamesArray[count] .setLayoutY(325-(lengthOfName*2));                    //setting the position for the labels
            labelForNamesArray[count] .setLayoutX(70+xPositionOfName-varyPosition);
            anchorPaneQueue.getChildren().addAll(labelForNamesArray[count]);
            xPositionOfName += 29;
        }
        int indexPosition = 0;
        for (int indexPassenger : passengerQueue.getUpdatedPassengersInQueue()) {      //this loop is to get the seatNumber and name of passenger in the queue and set it properly
            passengerButtonArray[indexPosition].setStyle("-fx-background-color:red; -fx-background-radius:40; -fx-border-radius:40; -fx-text-fill:white;");
            labelForNamesArray[indexPosition].setText(passengerQueue.getFirstName(indexPassenger));  //setting the name of the passenger to the train queue
            if(indexPassenger<10) {
```

```
            passengerButtonArray[indexPosition].setText("0" + indexPassenger);                    //setting
the text for the buttons
        }
        else{
            passengerButtonArray[indexPosition].setText(String.valueOf(indexPassenger));
        }
        indexPosition++;
    }
    buttonToTrainSeats.setOnAction(event1 -> {                              //when clicked it will change
the scene into the train seat layout
        Label detail = new Label("NOTE: Number on the seat represents the seat number in the
train.\n" +
                "          Red seat represent passenger on seat and grey \n          seat represent an empty
seat.");
        detail.setStyle("-fx-font-size:15px; -fx-font-weight:bold; -fx-font-style:italic;");
        detail.setLayoutX(700);
        detail.setLayoutY(230);
        stage.close();
        try {                                       // displaying the stage delay
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Button close = new Button("close");               //creates the close button and position it
properly
        close.setLayoutY(540);
        close.setLayoutX(1080);
        close.setMinWidth(60);
        close.setStyle("-fx-background-radius:20; -fx-border-radius:20; -fx-border-color:black; -fx-
border-width:2;");
```

```
        Label headingSeats = new Label("TRAIN SEATS");      //creates the heading label for this scene and positioning it properly
        headingSeats.setLayoutX(750);
        headingSeats.setLayoutY(100);
        headingSeats.setMinWidth(320);
        headingSeats.setAlignment(Pos.CENTER);
        headingSeats.setStyle(" -fx-text-fill:DARKBLUE; -fx-font-size:40px; -fx-background-radius:20; -fx-border-radius:20; " +
                "-fx-border-color:red; -fx-border-width:5; -fx-font-style:italic;");

        AnchorPane anchorPaneSeats = new AnchorPane();          //creating an anchorPane
        GridPane gridPaneSeats = new GridPane();                //creating a gridPane
        gridPaneSeats.setLayoutX(150);
        gridPaneSeats.setLayoutY(25);
        gridPaneSeats.setAlignment(Pos.CENTER);
        gridPaneSeats.setHgap(60);
        gridPaneSeats.setVgap(30);

        int colIndex = 0;                               //variables for the layout creation for the train seats
        int rowIndex = 0;
        int rowLoop = 0;
        int yPositionOfNameLabel = -20;
        int xPositionNameLabel = 0;
        for(int count=0; count<SEATING_CAPACITY; count++){                      //loop used to create labels and buttons for the train seats
            labelForNamesArray[count] = new Label("Empty seat");
            if (count<9){
                passengerButtonArray[count] = new Button("0"+(count+1));        //create new button with the seat number
            } else{
```

```
                passengerButtonArray[count] = new Button(String.valueOf(count+1));      //create new button with the seat number
            }
            labelForNamesArray[count].setLayoutX(120 + xPositionNameLabel);          //setting the position layouts
            labelForNamesArray[count].setLayoutY(yPositionOfNameLabel+25);
            passengerButtonArray[count].setStyle("-fx-background-radius:8; -fx-background-color:grey; -fx-text-fill:white;");
            passengerButtonArray[count].setMinSize(35,10);        //setting size
            gridPaneSeats.add(passengerButtonArray[count],colIndex,rowIndex);          //adding nodes to gridPane
            anchorPaneSeats.getChildren().addAll(labelForNamesArray[count]);
            colIndex++;                                          //incrementing
            if (colIndex==2){                                     //checks the condition and changes the value of colIndex
                colIndex+=2;
            }
            rowLoop++;
            xPositionNameLabel += 150;
            if(rowLoop==4){                                       //checks the condition and changes the value of the variables
                colIndex=0;
                rowLoop=0;
                rowIndex++;
                xPositionNameLabel = 0;
                yPositionOfNameLabel += 55;
            }
        }
        for(Passenger index : removedPassengerFromQueue){                    //loops this list to get the passengers to the train seats with their name
            passengerButtonArray[index.getSeatNumber()-1].setStyle("-fx-background-radius:8; -fx-background-color:red; -fx-text-fill:white; ");
```

```
            labelForNamesArray[index.getSeatNumber()-1].setText(index.getFirstName() +
index.getSurname());    //adding the name of the passenger
        }
        close.setOnAction(event2 -> {                                   //close button action
            closeProgram();
        });
        anchorPaneSeats.getChildren().addAll(close,headingSeats,gridPaneSeats,detail);  //adding the
nodes to the anchorPane
        anchorPaneSeats.setStyle("-fx-background-image:
url('File:/C:/Users/Nazhim/Desktop/pp2_cw2/src/sample/four.jpg');");   //backgroundImage for the
stage
        stage.setTitle("Viewing passengers");
        stage.setScene(new Scene(anchorPaneSeats, 1200, 620));   //creating and setting scene
        stage.show();
        stage.setOnCloseRequest(event2 -> {     //on close request
            event2.consume();
            closeProgram();
        });
    });

anchorPaneQueue.getChildren().addAll(counterLabel,tilePane,headingQueue,buttonToTrainSeats);
//adding the nodes to the anchorPane
    anchorPaneQueue.setStyle("-fx-background-image:
url('File:/C:/Users/Nazhim/Desktop/pp2_cw2/src/sample/two.jpg');");   //backgroundImage for the
stage
    stage.setTitle("Viewing passengers");
    stage.setScene(new Scene(anchorPaneQueue, 1275, 540));                              //creating
and setting scene
    stage.show();
    stage.setOnCloseRequest(event1 -> {       //on close request
        event1.consume();
        closeProgram();
```

```
        });
    });

anchorPaneWaiting.getChildren().addAll(gridPaneWaiting,heading,buttonToTrainQueue,noteForWaiting
);                  //adding the nodes to the anchorPane
    anchorPaneWaiting.setStyle("-fx-background-image:
url('File:/C:/Users/Nazhim/Desktop/pp2_cw2/src/sample/three.jpg');");  //backgroundImage for the
stage
    stage.setTitle("Viewing passengers");
    stage.setScene(new Scene(anchorPaneWaiting, 1200, 630));       //creating and setting scene
    stage.show();
    stage.setOnCloseRequest(event -> {       //on close request
        event.consume();
        closeProgram();
    });
  }
  public void deletePassengerFromTrainQueue(){                          //deleting a passenger from the
queue
    if(date != null && selectedRadioButton[0] != null) {                //checks if the user has selected a
date and a radio button firstly
        Scanner input = new Scanner(System.in);                         //creates the input scanner
        String firstName;                //variable for firstName
        String surname;                  //variable for Surname
        int seatNumber;                  //variable for SeatNumber
        System.out.println("\n===> Please enter the required details to delete the passenger from the
train queue. <==\n");    //display message
        System.out.print("-->\tEnter first name of the passenger: ");
        firstName = (input.nextLine()).trim();                                  //gets the users input
        while (!firstName.matches("[a-zA-Z_0-9]+")) {                   //validation check for the firstName
            System.out.println("\n  - You have entered an invalid name");
            System.out.print("-->\tEnter first name of the passenger: ");
```

```
            firstName = (input.nextLine()).trim();                              //gets the users input
        }

        System.out.print("-->\tEnter surname of the passenger: ");
        surname = " " + (input.nextLine()).trim();                          //gets the users surName input
        while (!surname.matches("[ a-z A-Z]+")) {                       //validation check for the Surname
            System.out.println("\n  - You have entered an invalid name");
            System.out.print("-->\tEnter surname of the passenger: ");
            surname = " " + (input.nextLine()).trim();                          //gets the users surName input
        }

        System.out.print("-->\tEnter seat number of the passenger: ");
        while (!input.hasNextInt()) {                                   //checks if the users input is a integer and
then loops if not a integer
            System.out.println("\n  - You have entered an invalid integer");
            System.out.print("-->\tEnter seat number of the passenger: ");
            input.nextLine();                                       //gets the users seatNumber
        }
        seatNumber = input.nextInt();                                   //sets the users seatNumber in to a
variable
        passengerQueue.deletingFromQueue(firstName.toLowerCase(), surname.toLowerCase(),
seatNumber);     //send the details to the passengerQueue class to delete
    } else{
        System.out.println("\nThere are no passengers in the train queue at the moment, so deletion of
passengers cannot take place.\n");
    }
    displayMenu();                                                  //calls the display menu method
  }
  public void  storeTrainQueueData(){
    try {
```

```
        if (dateCheck && radioCheck) {                                          //checks if the user has
selected a date and a radio button

            MongoClient mongoClient = new MongoClient("localhost", 27017);              // this is the
saving part

            DB databaseQueueSystem = mongoClient.getDB("trainQueue");                   //makes the
database

            DBCollection collection = databaseQueueSystem.getCollection("trainQueueCollection");
//creates th collection

            BasicDBObject document = new BasicDBObject();                               //creates the
document


            dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(dateSelected));            //storing
variables which are necessary to be loaded back

            dataToBeSavedIntoTheDatabaseCurrently.add(selectedRadioButton[0]);

            dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(newStartingPosition));

            dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(waitingArrayIndex));


dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(numberOfPassengersInWaitingRoomCurre
ntly));


            for (Passenger passenger : removedPassengerFromQueue) {                 //loops through
removedPassengerFromQueues and adds to the list

                dataToBeSavedIntoTheDatabaseCurrently.add(passenger.getFirstName());

                dataToBeSavedIntoTheDatabaseCurrently.add(passenger.getSurname());


dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passenger.getSecondsInQueue()));

                dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passenger.getSeatNumber()));

            }

            for (int index = removedPassengerFromQueue.size(); index < SEATING_CAPACITY; index++) {  //
the remaining part from the removedPassengerFromQueue made to null

                for (int count = 0; count < 4; count++) {

                    dataToBeSavedIntoTheDatabaseCurrently.add(null);

                }
```

```
        }
        for (int count = 0; count < sizeOfTheWaitingRoom; count++) {            // loops through waiting
room data and adds to the list
            dataToBeSavedIntoTheDatabaseCurrently.add(waitingRoom[count].getFirstName());
            dataToBeSavedIntoTheDatabaseCurrently.add(waitingRoom[count].getSurname());

dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(waitingRoom[count].getSecondsInQueue()
));

dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(waitingRoom[count].getSeatNumber()));
        }
        for (int count = sizeOfTheWaitingRoom; count < SEATING_CAPACITY; count++) {
            for (int index = 0; index < 4; index++) {
                dataToBeSavedIntoTheDatabaseCurrently.add(null);
            }
        }
        dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passengerQueue.getFirst()));
//loops through passengerQueue variables data and adds to the list
        dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passengerQueue.getLast()));

dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passengerQueue.getMaxLength()));
        dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passengerQueue.getSize()));

        for (int index : passengerQueue.getUpdatedPassengersInQueue()) {                //loops
through updatedPassengerInQueue and adds to the list
            dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(index));
        }
        for (int index = passengerQueue.getUpdatedPassengersInQueue().size(); index <
SEATING_CAPACITY; index++) {  //remaining slots in the updatedPassengerInQueue are made null
            dataToBeSavedIntoTheDatabaseCurrently.add(null);
        }
```

```
            for (int count = 0; count < (passengerQueue.getArrayPositionIndex()+1); count++) {      //loops through passengerQueue array and saves the data of passengers

dataToBeSavedIntoTheDatabaseCurrently.add(passengerQueue.getQueueArray()[count].getFirstName());

dataToBeSavedIntoTheDatabaseCurrently.add(passengerQueue.getQueueArray()[count].getSurname());

dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passengerQueue.getQueueArray()[count].getSecondsInQueue()));

dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(passengerQueue.getQueueArray()[count].getSeatNumber()));
            }
            for (int count = (passengerQueue.getArrayPositionIndex()+1); count < SEATING_CAPACITY; count++) { //if passengers aren't present in the queueArray, it's data is made to null
               for (int index = 0; index < 4; index++) {
                  dataToBeSavedIntoTheDatabaseCurrently.add(null);
               }
            }
            dataToBeSavedIntoTheDatabaseCurrently.add(String.valueOf(processDelayAlreadyCreated)); //saving the process delay part for option R
            for (int count = 0; count < dataToBeSavedIntoTheDatabaseCurrently.size(); count++) { //adds the list data into the document and inserts it
               document.append(String.valueOf(count), dataToBeSavedIntoTheDatabaseCurrently.get(count));
            }
            collection.insert(document);
            dataToBeSavedIntoTheDatabaseCurrently.clear();                        //clears the document at last
            System.out.println("Data successfully stored into the database.\n");
         } else {
            System.out.println("Sorry, there are no records to be saved into the database.\n");
```

```
        }
    }
    catch (Exception e){                    //exception handling
        System.out.println("Something went wrong when storing the data");
    }
    finally {
      displayMenu();
    }
  }
  public void clearDataBase(){              //this method will clear all the data stored in the database
    try {
        MongoClient mongoClient = new MongoClient("localhost", 27017);
        DB databaseQueueSystem = mongoClient.getDB("trainQueue");
        DBCollection collection = databaseQueueSystem.getCollection("trainQueueCollection");
        collection.drop();                  //this part clears the collection
        System.out.println("Database successfully cleared.\n");
    }catch (Exception e){                   //exception handling
        System.out.println("Something went wrong when clearing the database");
    }finally {
        displayMenu();
    }
  }
  public void loadTrainQueueData(){                 //this method is used to load the data back
    try {
        MongoClient mongoClient = new MongoClient("localhost", 27017);
        MongoDatabase database = mongoClient.getDatabase("trainQueue");         //gets the database by the name
        MongoCollection<Document> collection = database.getCollection("trainQueueCollection");
//gets the collection by the name
```

```
            loadedTrainQueueData.clear();
            for (Document dataSet : collection.find()) {                    //loops through the loaded data and inserts into the list
                loadedTrainQueueData.add(String.valueOf(dataSet));
            }
            System.out.println("Data successfully loaded from the database.");
        }
        catch (Exception e){                                    //exceptional handling part
            System.out.println("Something went wrong when loading the data");
        }
        finally {
            displayMenu();                                      //calls the display menu method
        }
    }
    public int timeDelay(){                                     //this method is used to create the time delay and return the time
        Random random = new Random();                           //creating the random variable
        int timeDelay = 0;
        for (int count = 0; count < 3; count++) {               //random dice loops 3 times
            int timeDelayDice = random.nextInt(6) + 1;      //range from 1 to 6
            timeDelay += timeDelayDice;                         //adds the timeDelay
        }
        return timeDelay;                                       //returns the time delay
    }
    public void setProcessingTimeForPassengers(){
        for(int index = 0; index< (passengerQueue.getArrayPositionIndex()+1); index++) {      // loops through the passenger in the trainQueue and adds the process delay
            passengerQueue.getQueueArray()[index].setSecondsInQueue(timeDelay());           //sets the time delay to the passengers
        }
```

```
      for(int index = (passengerQueue.getArrayPositionIndex()+1); index < sizeOfTheWaitingRoom; index++){  // loops through the passenger in the waitingRoom and adds the process delay

          waitingRoom[index].setSecondsInQueue(timeDelay());                    //sets the time delay to the passengers

      }

   }

   public void displayReport(){

      if(date != null && selectedRadioButton[0] != null) {                  //checks if the date and the radioButton are selected

          if(!processDelayAlreadyCreated){                                  //this runs to make the process delay for the passengers

              setProcessingTimeForPassengers();                             //calls this method

              processDelayAlreadyCreated = true;                            // this is used to stop setting new process delay every time when run

          }

          int loop = passengerQueue.getSize();                              //gets the size (number of passengers) of the trainQueue

          for(int index = 0; index < loop; index++) {                       // loops to remove all the passengers in the train queue

              Passenger removedPassenger = passengerQueue.remove();         //removing each passenger

              removedPassengerFromQueue.add(removedPassenger);              //adding the removed passenger into the removed passenger list

          }

          passengerQueue.setUpdatedPassengersInQueue();                     //updates the queue

          if(passengerQueue.getSize() == 0){                                //check if the queue is empty

              Alert informationBox = new Alert(Alert.AlertType.INFORMATION);     //creating a information box

              informationBox.setHeaderText("The train queue is empty");         //texts inside the information box

              informationBox.setTitle("Details");                           //setting a title to the information box

              informationBox.showAndWait();                                 //displays the alert box

          }
```

```
        AnchorPane anchorPane = new AnchorPane();                                   //creating an anchorPane

        anchorPane.setStyle("-fx-background-color:black;");                         //setting style

        Label report = new Label("__ R E P O R T __");                              //creating a label

        report.setStyle("-fx-font-size:30px; -fx-font-weight: bold; -fx-font-style:italic; -fx-text-fill:white;");
//setting style

        report.setLayoutX(540);                                                     //setting layout


        Label otherDetails = new Label();                   //creating label

        otherDetails.setLayoutY(570);                       //setting layout

        otherDetails.setLayoutX(700);                       //setting layout

        otherDetails.setStyle("-fx-font-size:14px; -fx-font-style:italic; -fx-text-fill:white;");         //setting
style

        otherDetails.setText("- Maximum length of the queue recorded : " +
passengerQueue.getMaxLength() + " passengers" + "       - Date : " + date  + "\n" +

                "- Maximum waiting time recorded : " + passengerQueue.getMaxStayInQueue() + "s"
+"                  - Journey : " + selectedRadioButton[0] +"\n" +

                "- Minimum waiting time recorded : " + passengerQueue.getMinStayInQueue() + "s" +
"\n" +

                "- Average waiting time recorded : " + passengerQueue.getAverageStayInQueue() + "s"
+ "\n");  //statistical details to display

        Label detailsOfPassenger = new Label("--Details of Passenger--");                   //creating label

        detailsOfPassenger.setStyle("-fx-font-size:15px; -fx-font-style:italic; -fx-text-fill:white;");
//setting style

        detailsOfPassenger.setLayoutY(20);                                          //setting layout

        detailsOfPassenger.setLayoutX(50);                                          //setting layout


        Label[] passengerDetails = new Label[SEATING_CAPACITY];  //creating passenger details label
array

        TilePane tilePane = new TilePane();                 //creating a tilePane

        tilePane.setLayoutY(50);                        //setting layout

        tilePane.setLayoutX(50);                        //setting layout

        tilePane.setMinSize(1200,800);     //setting the size
```

```
        for(int index = 0; index<SEATING_CAPACITY; index++){     //this loop is used to get the passenger details in the train queue and boarded passengers
            if(index>passengerQueue.getArrayPositionIndex()){
               break;
            }
            passengerDetails[index] = new Label();
            passengerDetails[index].setText("*Passenger " + (index+1) + "\n" +                    //details of passengers to be displayed
                  "First Name: " + passengerQueue.getQueueArray()[index].getFirstName() + "\n" +
                  "Surname:" + passengerQueue.getQueueArray()[index].getSurname() + "\n" +
                  "Waiting time in queue: " + passengerQueue.getQueueArray()[index].getSecondsInQueue() + "s\n" +
                  "Seat Number: " + passengerQueue.getQueueArray()[index].getSeatNumber() + "\n");
            passengerDetails[index].setMinSize(150,100);                    //setting size
            passengerDetails[index].setStyle("-fx-text-fill:white; -fx-font-style:italic;");          //setting style
            tilePane.getChildren().addAll(passengerDetails[index]);                          //adding the passenger details into the tilePane
        }
        File file = new File("Report" +" " +date +" "+selectedRadioButton[0]+ ".txt");       //this part is used to create the text file
        FileWriter fileWriter;
        PrintWriter printWriter;
        try{
            fileWriter = new FileWriter(file);                                  //creating fileWriter to add the file
            printWriter = new PrintWriter(fileWriter,true);                      //creating printWriter to write into the file
            printWriter.println("\n ___REPORT___\n\n  Details of the passengers\n");
            printWriter.println(passengerQueue.display());                                  //write the data in.
            printWriter.println("\n\n  Maximum length of the queue recorded : " + passengerQueue.getMaxLength() + " passengers" + "\n" +
```

```
                "  Maximum waiting time recorded : " + passengerQueue.getMaxStayInQueue() + "s" +"\n"
+
                "  Minimum waiting time recorded : " + passengerQueue.getMinStayInQueue() + "s" + "\n"
+
                "  Average waiting time recorded : " + passengerQueue.getAverageStayInQueue() + "s" +
"\n"+
                "  Date : " + date  + "\n" +
                "  Journey : " + selectedRadioButton[0] +"\n");
            fileWriter.close();                                    //closes the file
            printWriter.close();                                    //closes the file
        }
        catch (FileNotFoundException e){                                //exception handling
            System.out.println("file not found");
        } catch (IOException e) {                                   //exception handling
            System.out.println("No permission to write the file");
        }
        anchorPane.getChildren().addAll(report,otherDetails,detailsOfPassenger,tilePane);       //adding
the nodes in to the anchorPane
        stage.setScene(new Scene(anchorPane, 1300, 660));                   //creating and setting the
scene into the stage
        stage.setTitle("Detailed Report");
        stage.show();
        stage.setOnCloseRequest(event -> {                                  //when closing the stage
            event.consume();                                        //consumes the event
            closeProgram();                                         //calls the close program
        });
    }else{
        System.out.println("\nThere is no date and journey selected, therefore report cannot be
generated. \n");
        displayMenu();
    }
```

```
    }
    public void closeProgram(){
        Alert confirmationBox = new Alert(Alert.AlertType.CONFIRMATION);        //creating the
confirmation box
        confirmationBox.setTitle("Close program");                          //setting title for the confirmation box
        confirmationBox.setHeaderText("Are you sure you want to close ?");      //setting header for the
confirmation box
        ButtonType yes = new ButtonType("Yes");                             //creating the button YES
        ButtonType no = new ButtonType("No");                               //creating the button NO
        confirmationBox.getButtonTypes().setAll(yes, no);
        Optional<ButtonType> result = confirmationBox.showAndWait();          //displays the confirmation
box
        if (result.get() == yes){                                   //checks if the user has clicked YES
            stage.close();                                        //closes the window
            displayMenu();                                        //calls the displayMenu method
        }
    }
    public static void main(String[] args) {                              //main method
        launch(args);
    }
}
```

```
//passenger class
// w1761265  SE2019281  Mohammed Nazhim Kalam
package sample;
public class Passenger {
   private String firstName,surname;                                    //firstName variable and surname variable of string type
   private int secondsInQueue,seatNumber;                               //waiting time and seatNumber variable of integer type

   public int getSeatNumber(){                                          //getter for seatNumber
      return seatNumber;
   }

   public void setSeatNumber(int seatNumber){                           //setter for the seatNumber
      this.seatNumber = seatNumber;
   }

   public String getFirstName(){                                        //getter for the firstName
      return firstName;
   }

   public String getSurname(){                                          //getter for the surname
      return surname;
   }

   public int getSecondsInQueue(){                                      //getter for the waiting time in queue
      return secondsInQueue;
   }
```

```
    public void setFirstName(String firstName){                    //setter for the firstName
        this.firstName = firstName;
    }

    public void setSurname(String surname){                        //setter for the surname
        this.surname = " " + surname.trim();
    }

    public void setSecondsInQueue(int secondsInQueue){             //setter for the waiting time in queue
        this.secondsInQueue = secondsInQueue;
    }
}
```

```
//passengerQueue class
// w1761265  SE2019281  Mohammed Nazhim Kalam
package sample;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import static sample.TrainStation.SEATING_CAPACITY;
import static sample.TrainStation.waitingArrayIndex;

public class PassengerQueue {
   private Passenger[] queueArray = new Passenger[SEATING_CAPACITY];     //this is an array with the passengers
   private int first,last;                                  //the first about to leave and last person entered in the queue
   private int maxStayInQueue,minStayInQueue;                     //the maximum and minimum time spent in the queue
   private double averageStayInQueue;                          //records the average time spent by a passenger in the queue.
   private int maxLength,size;                                //maximum length of the queue attained and the size of the queue
   private int arrayPositionIndex = -1;                          //queue array position
   public List<Integer> updatedPassengersInQueue = new ArrayList<>();    //create a list to get the updated Passengers In Queue

   public int getArrayPositionIndex(){                                                  //getter for arrayPositionIndex
      return arrayPositionIndex;
   }

   public void setArrayPositionIndex(int value){                                        //setter for setArrayPositionIndex
      arrayPositionIndex = value;
```

```
    }

    public int getMinStayInQueue() {                                    //getter for minStayInQueue
        return minStayInQueue;
    }

    public double getAverageStayInQueue(){                              //getter for averageStayInQueue
        return averageStayInQueue;
    }


    public void setQueueArray(Passenger passenger,int position){        //setter for setQueueArray
        queueArray[position] = passenger;
    }

    public Passenger[] getQueueArray(){                                 //getter for queueArray
        return queueArray;
    }

    public void setFirst(int first){                                    //setter for setFirst
        this.first = first;
    }

    public int getFirst(){                                              //getter for first
        return first;
    }
```

```
  public void setLast(int last){                                    //setter for setLast
    this.last = last;
  }

  public int getLast(){                                             //getter for last
    return last;
  }

  public void setSize(int size){                                    //setter for setSize
    this.size = size;
  }

  public int getSize(){                                             //getter for size
    return size;
  }


  public void add(Passenger passenger){          //adding the passenger to the queue
    if(!isFull()) {                        //if not full we can add elements
      queueArray[last] = passenger;          //at start rear=0, and we add data to it
      last = (last + 1)%SEATING_CAPACITY;     //updating rear value for circular array (we start from position one)
      size++;                          //when data entered size also increases by one
      arrayPositionIndex++;                //position of the element added in the queueArray
    }
  }

  public Passenger remove(){                    //this is used to remove the passenger from the queue
```

```
    Passenger removedPassenger = queueArray[first];     //we get the data from the queue at the front position
    if(!isEmpty()) {                            //if not empty we can remove data
      first = (first + 1) % SEATING_CAPACITY;        //updating front value for circular array
      size = size - 1;                          //after removing the size of the array will also reduce by 1
    }
    return removedPassenger;                        //if the removed data is needed to be used in the program
  }

  public void deletingFromQueue(String firstName, String surname, int seatNumber){
    boolean valid = false;
    for (int index = 0; index < size; index++) {                                    //loops the queueArray elements
      if (queueArray[(first + index) % SEATING_CAPACITY].getFirstName().equals(firstName)) { //checks if the firstName entered is equal
        if (queueArray[(first + index) % SEATING_CAPACITY].getSurname().equals(surname)) { //checks if the surname entered is equal
          if (queueArray[(first + index) % SEATING_CAPACITY].getSeatNumber() == seatNumber) { //checks if the seatNumber entered is equal
            System.out.println("\nPassenger is successfully deleted from the train queue\n"); //displays message
            System.out.println("Details of the deleted passenger includes : ");
            System.out.println("-----------------------------------------------------------");
            System.out.println("\t First name: " + queueArray[(first + index) % SEATING_CAPACITY].getFirstName());
            System.out.println("\t Surname:" + queueArray[(first + index) % SEATING_CAPACITY].getSurname());
            System.out.println("\t Seat number booked : " + queueArray[(first + index) % SEATING_CAPACITY].getSeatNumber());
            for (int count = index; count < size; count++) {          //reordering the queueArray and pushing the deleted element to the last position
              if (count != (size - 1)) {                    //if it's not the last element
```

```
                    queueArray[(first + count) % SEATING_CAPACITY] = queueArray[(first + count  + 1) %
SEATING_CAPACITY];
                  }
                }
                if(maxLength==size){          //checks if the maxLength was calculated with the deleted
item
                  maxLength--;              //decrements the max length
                }
                size--;                     //reduces the size by one
                last = arrayPositionIndex;      //make new last position
                arrayPositionIndex--;          //decrements the array index
                valid = true;
                waitingArrayIndex--;          //this is to say that a passenger is removed to the number of
people in total has to reduce by one
                setUpdatedPassengersInQueue();   //calls the setUpdatedPassengerInQueue
              }
            }
          }
        }
        if (!valid) {
          System.out.println("-----------------------------------------------------------------");
          System.out.println("\tThere is no passenger in the queue with the given details above");
        }
    }

    public void setMaxLength(int length){        //we set the max length of the queue, when adding
passengers, we call this method to set the size accordingly.
      if(length>maxLength) {                //checks if the length is greater than the max length and changes
accordingly
        maxLength = length;
```

```
    }
  }

  public int getMaxLength(){  //getting the maximum length of the queue
    return maxLength;
  }

  public void setMaxStayInQueueAndSetMinStayInQueue(){
    List<Integer> times = new ArrayList<>();                    //gets all the times of the passengers in the train queue and boarded ones.
    double total = 0;
    for (int index = 0; index <= arrayPositionIndex; index++) {          // loops to add the waiting times of the passengers into a list
      times.add(queueArray[index].getSecondsInQueue());
      total += queueArray[index].getSecondsInQueue();                   //adds the time for average calculation
    }
    if(times.size()!=0) {
      maxStayInQueue = Collections.max(times);                    //get the maximum time in the list
      minStayInQueue = Collections.min(times);                    //get the minimum time in the list
      double average = Math.round(total / (arrayPositionIndex + 1) * 100);      //gets the average time for each passenger into 2dp
      averageStayInQueue = average / 100;                         //sets the average value
    }
  }

  public int getMaxStayInQueue(){   //getting the total max size of the queue
    return maxStayInQueue;
  }
```

```
  public boolean isEmpty(){     //this method returns "true" if the size is equal to 0
    return getSize() == 0;
  }

  public boolean isFull(){          //this method returns "true" if the size is equal to 42
    return getSize() == SEATING_CAPACITY;
  }

  public String display(){                          //this method is used to display the content into the text file
    StringBuilder stringBuilder = new StringBuilder();
    for(int index = 0; index<=arrayPositionIndex; index++){
      stringBuilder.append("  *Passenger ").append(index+1).append("\n").append("   -First Name: " + queueArray[index].getFirstName() + "\n");
      stringBuilder.append("   -Surname:").append(queueArray[index].getSurname()).append("\n").append("   -Waiting time in queue: " + queueArray[index].getSecondsInQueue() + "s\n");
      stringBuilder.append("   -Seat Number: ").append(queueArray[index].getSeatNumber()).append("\n\n");

    }
    return stringBuilder.toString();              //returns the string
  }

  public void setUpdatedPassengersInQueue(){          // this method is used to get the seatNumber which represents the passengers in the train queue currently
    try {
      updatedPassengersInQueue.clear();           //clears the list
      for (int index = 0; index < size; index++) {   //loops and add the seat number into the list
```

```
            updatedPassengersInQueue.add(queueArray[(first + index) %
SEATING_CAPACITY].getSeatNumber());
        }
        setMaxLength(updatedPassengersInQueue.size());    //calls this method to set the size
        setMaxStayInQueueAndSetMinStayInQueue();         //calls this method to set the times
      }catch (Exception e){
        System.out.println("Something went wrong when updating the updatedPassengersInQueue list");
      }
   }


   public List<Integer> getUpdatedPassengersInQueue(){
      return updatedPassengersInQueue;
   }


   public String getFirstName(int seatNumber){
      String FirstNameCustomer = "";            //this method is used to send the firstName only to the train
queue in view method
      for(int count = 0; count<size; count++){
        if(queueArray[(first+count)%SEATING_CAPACITY].getSeatNumber()==seatNumber){
           FirstNameCustomer = queueArray[(first+count)%SEATING_CAPACITY].getFirstName();
//extracts the first name of the passenger and send
        }
      }
      return  FirstNameCustomer;                //returns the first name
   }
}
```

TEST PLAN for Programming Principles 2 (Train Queue program CW02). W1761265

In order to test CW02 program functionality (train queue) we have to enter data to CW01 program (train booking)

CW01 Implementation

- Book seat 1 to 10 with the customer name "Nazhim Kalam" for date 2020-06-11 (NOT RETURN JOURNEY).
  Book seat 11 to 20 with the customer name "Hasini Gomez" for date 2020-06-11 (NOT RETURN JOURNEY).
  Book seat 21 to 30 with the customer name "Niveda Thomas" for date 2020-06-11 (NOT RETURN JOURNEY).
  Book seat 31 to 42 with the customer name "Namith Nimilaka" for date 2020-06-11 (NOT RETURN JOURNEY).

- Book seat 1 to 10 with the customer name "Ramesh Hetti" for date 2020-06-11 (RETURN JOURNEY).
  Book seat 11 to 20 with the customer name "Suresh Navin" for date 2020-06-11 (RETURN JOURNEY).
  Book seat 21 to 30 with the customer name "Mahesh Khan" for date 2020-06-11 (RETURN JOURNEY).
  Book seat 31 to 42 with the customer name "Arujun Reddy" for date 2020-06-11 (RETURN JOURNEY).

(you can enter the option in uppercase or lowercase)

Student Name: Nazhim Kalam		**Student ID: 2019281**	**UoW ID: w1761265**	
Test No:	**Test Input**	**Expected Result**	**Actual Result**	**Pass/ Fail**
1	* Enter option: A * Click the exit button "RED CROSS" button and click "YES" button	Displays a GUI requesting the user to select a date from the date picker and the journey type from the radio buttons and disappears when exited.	Displays a GUI requesting the user to select a date from the date picker and the journey type from the radio buttons and disappears when exited.	Pass
2	*Enter option: V *Click the button "click for train queue" *Click the button "click for train seats" *Click the "close" button *Click the "YES" button	Displays the WAITING ROOM GUI, displays the TRAIN QUEUE GUI and displays the TRAIN SEATS GUI. GUI disappears when exited.	Displays the WAITING ROOM GUI, displays the TRAIN QUEUE GUI and displays the TRAIN SEATS GUI. GUI disappears when exited.	Pass

3	*Enter option: D	Displays message “There are no passengers in the train queue at the moment, so deletion of passengers cannot take place.”	Displays message “There are no passengers in the train queue at the moment, so deletion of passengers cannot take place.”	Pass
4	Enter option: S	Displays message “Sorry, there are no records to be saved into the database.”	Displays message “Sorry, there are no records to be saved into the database.”	Pass
5	Enter option: L	Displays message “Data successfully loaded from the database.”	Displays message “Data successfully loaded from the database.”	Pass
6	Enter option: C	Displays message “Database successfully cleared.”	Displays message “Database successfully cleared.”	Pass
7	Enter option: R	Displays message “There is no date and journey selected, therefore report cannot be generated.”	Displays message “There is no date and journey selected, therefore report cannot be generated.”	Pass
8	Enter option: Q	Displays message “Thanks for visiting Denuwara Express. Program exiting...”	Displays message “Thanks for visiting Denuwara Express. Program exiting...”	Pass

Don’t re-run program for each test case from (9 to 18)

9	*Enter option: A *Select 2020-06-11 from the date picker. *Select “Not return journey” radio button. *Click OK button *Enter “y” for all the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button.	GUI for selecting date and the journey radio button will be displayed firstly. Displays the list of passengers who checked into the waiting room. Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	GUI for selecting date and the journey radio button will be displayed firstly. Displays the list of passengers who checked into the waiting room. Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Pass
10	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the train queue GUI indicating who are present in the train queue with their names.	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the train queue GUI indicating who are present in the train queue with their names.	Pass

		Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	
11	*Enter option: d *Enter first name: “nazhim” *Enter surname: “kalam” *Enter seat number: 6 or (any available seat in the train queue within the range 1 to 10)	Display message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: nazhim Surname: kalam Seat number booked : 6 ”	Display message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: nazhim Surname: kalam Seat number booked : 6 ”	Pass
12	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the updated train queue (you won’t able to see the delete passenger from the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the updated train queue (you won’t able to see the delete passenger from the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Pass
13	*Enter option: R *Close the displayed GUI using the RED CROSS (close) button.	Displays an ALERT BOX with the message “The train queue is empty” Displays the REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Displays an ALERT BOX with the message “The train queue is empty” Displays the REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Pass
14	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the	Pass

		the train queue to the train seats with their names. GUI disappears when closed	train queue to the train seats with their names. GUI disappears when closed	
15	*Enter option: A *Close the displayed GUI using the RED CROSS (close) button.	Displays the updated GUI (new set of passengers are added from the waiting room to the train queue). GUI disappears when closed	Displays the updated GUI (new set of passengers are added from the waiting room to the train queue). GUI disappears when closed	Pass
16	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) of who are present in the waiting room with their names. Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) of who are present in the waiting room with their names. Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Pass
17	*Enter option: R *Close the displayed GUI using the RED CROSS (close) button.	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Pass
18	*Enter option: S *Enter option: Q	Displays message, “Data successfully stored into the database.” “Thanks for visiting Denuwara Express. Program exiting...” and exits the program	Displays message, “Data successfully stored into the database.” “Thanks for visiting Denuwara Express. Program exiting...” and exits the program	Pass

Re-run the program

19	*Enter option: L	Display message “Data successfully loaded from the database.”	Display message “Data successfully loaded from the database.”	Pass
20	*Enter option: A *Select 2020-06-11 from the date picker. *Select “Not return journey” radio button. *Click OK button.	Displays the loaded train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Displays the loaded train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Pass
21	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Pass
22	*Enter option: D (depending on the passengers in the train queue) *Enter first name: hasini *Enter surname: Gomez *Enter seat number: 15 or (any available seat in the train queue within the range 11 to 20)	Displays message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: hasini Surname: gomez Seat number booked : 15”	Displays message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: hasini Surname: gomez Seat number booked : 15”	Pass
23	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that the deleted passenger is not found in the train queue) GUI indicating who are	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that the deleted passenger is not found in the train queue) GUI	Pass

		present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	
24	*Enter option: R *Close the displayed GUI using the RED CROSS (close) button.	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Pass
25	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the train queue to the train seats with their names. GUI disappears when closed	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the train queue to the train seats with their names. GUI disappears when closed	Pass
26	*Enter option: A *Click on the RED CROSS (close) button	Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Pass
27	*Enter option: S *Enter option: Q	Displays message, “Data successfully stored into the database. Thanks for visiting Denuwara Express. Program exiting...”	Displays message, “Data successfully stored into the database. Thanks for visiting Denuwara Express. Program exiting...”	Pass

Return Journey

Don’t re-run program for test cases from (28 to 37)

28	*Enter option: A *Select 2020-06-11 from the date picker. *Select “return journey” radio button. *Click OK button *Enter “y” for the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button.	GUI for selecting date and the journey radio button will be displayed firstly. Displays the list of passengers who checked into the waiting room. Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	GUI for selecting date and the journey radio button will be displayed firstly. Displays the list of passengers who checked into the waiting room. Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Pass
29	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the train queue GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the train queue GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Pass
30	*Enter option: d *Enter first name: “ramesh” *Enter surname: “hetti” *Enter seat number: 6 or (any available seat in the train queue within the range 1 to 10)	Display message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: ramesh Surname: hetti Seat number booked : 6 ”	Display message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: ramesh Surname: hetti Seat number booked : 6 ”	Pass
31	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button	Displays the waiting room GUI indicating who are present in the waiting room with their names.	Displays the waiting room GUI indicating who are present in the waiting room with their names.	Pass

	*Click the “close” button *Select “yes” to close	Displays the updated train queue (you won’t able to the delete passenger from the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the updated train queue (you won’t able to the delete passenger from the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	
32	*Enter option: R *Close the displayed GUI using the RED CROSS (close) button.	Displays an ALERT BOX with the message “The train queue is empty” Displays the REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Displays an ALERT BOX with the message “The train queue is empty” Displays the REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Pass
33	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the train queue to the train seats with their names. GUI disappears when closed	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the train queue to the train seats with their names. GUI disappears when closed	Pass
34	*Enter option: A *Close the displayed GUI using the RED CROSS (close) button.	Displays the updated GUI (new set of passengers are added from the waiting room to the train queue) of the train queue. GUI disappears when closed	Displays the updated GUI (new set of passengers are added from the waiting room to the train queue) of the train queue. GUI disappears when closed	Pass
35	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that a new set of passengers have been added into the	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue)	Pass

		train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	
36	*Enter option: R *Close the displayed GUI using the RED CROSS (close) button.	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Pass
37	*Enter option: S *Enter option: Q	Displays message, “Data successfully stored into the database.” “Thanks for visiting Denuwara Express. Program exiting...” and exits the program	Displays message, “Data successfully stored into the database.” “Thanks for visiting Denuwara Express. Program exiting...” and exits the program	Pass

Re-run the program

38	*Enter option: L	Display message “Data successfully loaded from the database.”	Display message “Data successfully loaded from the database.”	Pass
39	*Enter option: A *Select 2020-06-11 from the date picker. *Select “return journey” radio button. *Click OK button.	Displays the loaded train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Displays the loaded train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Pass
40	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) indicating who are present in the waiting room with their names.	Displays the updated waiting room GUI (you will notice that a number of passengers have been removed from the waiting room) indicating who are present in the waiting room with their names.	Pass

		Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the updated train queue (you will notice that a new set of passengers have been added into the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	
41	*Enter option: D (depending on the passengers in the train queue) *Enter first name: suresh *Enter surname: navin *Enter seat number: 15 or (any available seat in the train queue within the range 11 to 20)	Displays message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: suresh Surname: navin Seat number booked : 15”	Displays message, “Passenger is successfully deleted from the train queue Details of the deleted passenger includes: ---------- First name: suresh Surname: navin Seat number booked : 15”	Pass
42	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that the deleted passenger is not found in the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays the updated train queue (you will notice that the deleted passenger is not found in the train queue) GUI indicating who are present in the train queue with their names. Displays the train seat GUI indicating who are boarded in to train with their names against their respective seat GUI disappears when closed	Pass
43	*Enter option: R *Close the displayed GUI using the RED CROSS (close) button.	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Displays an ALERT BOX with the message “The train queue is empty” Displays the updated REPORT GUI with the required details and also produces a text file of the report. GUI disappears when closed	Pass

44	* Enter option: V *Click the “click for train queue” button *Click the “click for train seats” button *Click the “close” button *Select “yes” to close	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the train queue to the train seats with their names. GUI disappears when closed	Displays the waiting room GUI indicating who are present in the waiting room with their names. Displays an empty train queue GUI Displays the updated train seat GUI with all the passenger moved from the train queue to the train seats with their names. GUI disappears when closed	Pass
45	*Enter option: A *Click on the RED CROSS (close) button	Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Displays the train queue GUI, with the random number of passengers moved from the waiting room to the train queue.	Pass
46	*Enter option: S *Enter option: Q	Displays message, “Data successfully stored into the database. Thanks for visiting Denuwara Express. Program exiting...”	Displays message, “Data successfully stored into the database. Thanks for visiting Denuwara Express. Program exiting...”	Pass

VALIDATIONS (re run the program for each test case below)

47	*Enter option: A *Select date 2020-06-20 *Select the return journey radio button *Click OK button	Displays an Alert box with the message, “Please select a different date because there are no bookings made on this date”	Displays an Alert box with the message, “Please select a different date because there are no bookings made on this date”	Pass
48	*Enter option: A *Select date 2020-06-11 *Select the return journey radio button. *Click the OK button *For the check in details for a passenger Enter any other letter other than “y” and “n’	Displays the message “Please enter only "Y" or "N" only.” And asks the user to re enter	Displays the message “Please enter only "Y" or "N" only.” And asks the user to re enter	Pass
49	*Enter option: A *Click the OK button	Displays an Alert box with the message, “Please select a radio button and select a date from the date picker”	Displays an Alert box with the message, “Please select a radio button and select a date from the date picker”	Pass
50	*Enter option: A *select any date from the date picker. *Click the OK button	Displays an Alert box with the message “Please select a radio button”	Displays an Alert box with the message “Please select a radio button”	Pass
51	*Enter option: A	Displays an Alert box with the message “Now the train queue is full”	Displays an Alert box with the message “Now the train queue is full”	Pass

	*Select 2020-06-11 from the date picker. *Select “Not return journey” radio button. *Click OK button *Enter “y” for all the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button. *Keep entering option: A until all 42 passengers get into the train queue			
52	During transferring passengers from waiting room to the train queue under option A, if the transfer number is greater than the number of passengers in the waiting room.	Displaying an Alert box indicating that the number of the passengers in the waiting room is less than the transfer amount number, therefore the number of passengers in the waiting room will be transferred to the train queue.	Displaying an Alert box indicating that the number of the passengers in the waiting room is less than the transfer amount number, therefore the number of passengers in the waiting room will be transferred to the train queue.	Pass
53	Continuation from the testcase ***51***, *Enter option: R	Displays an Alert box with the message, “The train queue is empty”	Displays an Alert box with the message, “The train queue is empty”	Pass
54	*Enter option: A *Select any radio button *Select OK button	Displays an Alert box with the message, “Please select a date from the date picker”	Displays an Alert box with the message, “Please select a date from the date picker”	Pass
55	*Enter option: A *Select 2020-06-11 from the date picker. *Select “Not return journey” radio button. *Click OK button *Enter “y” for all the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button. *Enter option: D *Enter first name:!@#$%^&*()	Displays message and asks the user to re-enter the first name, “You have entered an invalid name Enter first name of the passenger:”	Displays message and asks the user to re-enter the first name, “You have entered an invalid name Enter first name of the passenger:”	Pass

56	*Enter option: A *Select 2020-06-11 from the date picker. *Select “Not return journey” radio button. *Click OK button *Enter “y” for all the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button. *Enter option: D *Enter first name: “Nazhim” *Enter surname:”!@#$%)45613”	Displays message and asks the user to re-enter the Surname, “You have entered an invalid name Enter surname of the passenger:”	Displays message and asks the user to re-enter the Surname, “You have entered an invalid name Enter surname of the passenger:”	Pass
57	*Enter option: A *Select 2020-06-11 from the date picker. *Select “Not return journey” radio button. *Click OK button *Enter “y” for all the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button. *Enter option: D *Enter first name: “Nazhim” *Enter surname: “kalam” *Enter seat number: “jabdv”	Displays message and asks the user to re-enter the seat number, “You have entered an invalid integer Enter seat number of the passenger:”	Displays message and asks the user to re-enter the seat number, “You have entered an invalid integer Enter seat number of the passenger:”	Pass
58	When user clicks the red cross (exit) button or any close button in the GUI	Displays a confirmation box asking the user if you are sure to close or not.	Displays a confirmation box asking the user if you are sure to close or not.	Pass
59	When there are no passengers to be moved from the waiting room to the train queue by enter the option A: repeatedly	Displays an ALERT BOX with the message “The waiting room is empty, therefore there are no passengers to be moved into the train queue”	Displays an ALERT BOX with the message “The waiting room is empty, therefore there are no passengers to be moved into the train queue”	Pass
60	*Enter option: A *Select 2020-06-11 from the date picker.	Displays message “There is no passenger in the queue with the given details above’	Displays message “There is no passenger in the queue with the given details above’	Pass

	*Select “Not return journey” radio button. *Click OK button *Enter “y” for all the customers for checking into the waiting room. *Close the displayed GUI using the RED CROSS (close) button. *Enter option: D *Enter first name: “Hashim” *Enter surname: “kalam” *Enter seat number: “10”			
61	Enter option: Z	Displays a message “You have entered an incorrect option... Please enter a correct option from (A,V,D,S,L,C,R,Q) only...”	Displays a message “You have entered an incorrect option... Please enter a correct option from (A,V,D,S,L,C,R,Q) only...”	Pass

ADDITIONAL

62	*Enter option: C	Displays the message, “Database successfully cleared.” This deletes all records in the database	Displays the message, “Database successfully cleared.” This deletes all records in the database	Pass