UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY
In collaboration with
UNIVERSITY OF WESTMINSTER
Object Oriented Principles
5COSC007C

Coursework – Phase 3

Vehicle Rental System

Module Leader's Name – Mr. Guhanathan Poravi

Dinuka Piyadigama
UoW ID – 17421047
IIT ID – 2018373

Contents

GUI

Code

```
package lk.dinuka.VehicleRentalSystem.View;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import lk.dinuka.VehicleRentalSystem.Controller.DatabaseController;
import lk.dinuka.VehicleRentalSystem.Controller.GUIController;
import lk.dinuka.VehicleRentalSystem.Controller.WestminsterRentalVehicleManager;
import lk.dinuka.VehicleRentalSystem.Model.*;

import java.util.ArrayList;

public class GUI extends Application {
   public static void main(String[] args) {
      launch(args);
   }

   private static ArrayList<Vehicle> searchedVehicles = new ArrayList<>();        //used to pass in
searched vehicles into the table
   private static ArrayList<Vehicle> searchInSearch = new ArrayList<>();      //used to filter
search by Vehicle type


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

   @Override
   public void start(Stage primaryStage) throws Exception {
//       Platform.setImplicitExit(false);
      primaryStage.setTitle("List of vehicles in system");

      TableView tableOfVehicles = new TableView();

      //Creating columns to be added to the table
      TableColumn<String, Vehicle> plateNoColumn = new TableColumn<>("Plate No");
      plateNoColumn.setCellValueFactory(new PropertyValueFactory<>("plateNo"));

      TableColumn<String, Vehicle> makeColumn = new TableColumn<>("Make");
      makeColumn.setCellValueFactory(new PropertyValueFactory<>("make"));
```

```
        TableColumn<String, Vehicle> modelColumn = new TableColumn<>("Model");
        modelColumn.setCellValueFactory(new PropertyValueFactory<>("model"));

        TableColumn<String, Vehicle> availabilityColumn = new TableColumn<>("Availability");
        availabilityColumn.setCellValueFactory(new PropertyValueFactory<>("availability"));
        availabilityColumn.setMinWidth(100);

        TableColumn<String, Vehicle> engineCapacityColumn = new TableColumn<>("Engine
Capacity");
        engineCapacityColumn.setCellValueFactory(new
PropertyValueFactory<>("engineCapacity"));
        engineCapacityColumn.setMinWidth(130);

        TableColumn<String, Vehicle> dailyCostColumn = new TableColumn<>("Daily Cost");
        dailyCostColumn.setCellValueFactory(new PropertyValueFactory<>("dailyCost"));
        dailyCostColumn.setMinWidth(110);

        TableColumn<String, Vehicle> typeColumn = new TableColumn<>("Type");
        typeColumn.setCellValueFactory(new PropertyValueFactory<>("type"));

        TableColumn<String, Vehicle> transmissionColumn = new TableColumn<>("Transmission");
        transmissionColumn.setCellValueFactory(new PropertyValueFactory<>("transmission"));
        transmissionColumn.setMinWidth(130);

        TableColumn<String, Vehicle> hasAirConColumn = new TableColumn<>("Has Air
Conditioning");
        hasAirConColumn.setCellValueFactory(new PropertyValueFactory<>("hasAirCon"));
        hasAirConColumn.setMinWidth(180);

        TableColumn<String, Vehicle> startTypeColumn = new TableColumn<>("Start Type");
        startTypeColumn.setCellValueFactory(new PropertyValueFactory<>("startType"));
        startTypeColumn.setMinWidth(120);

        TableColumn<String, Vehicle> wheelSizeColumn = new TableColumn<>("Wheel Size");
        wheelSizeColumn.setCellValueFactory(new PropertyValueFactory<>("wheelSize"));
        wheelSizeColumn.setMinWidth(130);


        tableOfVehicles.getColumns().addAll(plateNoColumn, makeColumn, modelColumn,
availabilityColumn, engineCapacityColumn, dailyCostColumn,
                typeColumn, transmissionColumn, hasAirConColumn, startTypeColumn,
wheelSizeColumn);           //adding all the columns to the table


tableOfVehicles.getItems().addAll(WestminsterRentalVehicleManager.getVehiclesInSystem());
//adding all the vehicles in the available
        // in the vehiclesInSystem ArrayList
```

```
        searchedVehicles.addAll(WestminsterRentalVehicleManager.getVehiclesInSystem());
    //to get filter by vehicle type to work before searching for a Make

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

        HBox searchSection = new HBox();
        searchSection.setMinWidth(220);
        searchSection.getChildren().add(new Label("Search Make:"));

        TextField makeSearch = new TextField();
        searchSection.getChildren().add(makeSearch);

        Button searchClick = new Button("Search");
        searchSection.getChildren().add(searchClick);
        Button resetClick = new Button("Reset");
        searchSection.getChildren().add(resetClick);


    //      VBox filterSection = new VBox(new Label("Filter By"));

        HBox filterType = new HBox(new Label("Filter Type:"));
        Button filterCarClick = new Button("Cars");
        filterType.getChildren().add(filterCarClick);
        Button filterBikeClick = new Button("Motorbikes");
        filterType.getChildren().add(filterBikeClick);

        filterType.setPadding(new Insets(10, 0, 0, 0));

    //      HBox filterEngineCap = new HBox(new Label("Engine Capacity:"));


        filterType.setMinWidth(200);
    //      filterSection.getChildren().addAll(filterType);

        VBox allSearchFilter = new VBox(searchSection, filterType);
        allSearchFilter.setPadding(new Insets(20, 0, 20, 20));

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

        VBox bookingSection = new VBox();

        HBox allDates = new HBox();

        //pick up date entry section
        HBox pickUpDateSec = new HBox(new Label("Pick Up:"));
        TextField dayPickUp = new TextField();
        TextField monthPickUp = new TextField();
        TextField yearPickUp = new TextField();
```

```
        dayPickUp.setPrefWidth(40);
        monthPickUp.setPrefWidth(40);
        yearPickUp.setPrefWidth(80);

        pickUpDateSec.getChildren().addAll(dayPickUp, monthPickUp, yearPickUp);

        //drop off date entry section
        HBox dropOffDateSec = new HBox();
        Label dropOffLabel = new Label("Drop Off:");

        TextField dayDropOff = new TextField();
        TextField monthDropOff = new TextField();
        TextField yearDropOff = new TextField();

        dayDropOff.setPrefWidth(40);
        monthDropOff.setPrefWidth(40);
        yearDropOff.setPrefWidth(80);

        dropOffDateSec.getChildren().addAll(dropOffLabel, dayDropOff, monthDropOff,
yearDropOff);


        Button availabilityCheck = new Button("Check Availability");

        allDates.setSpacing(10.0);

        Button bookOnClick = new Button("Book");
//        bookOnClick.setAlignment(right);

        Text checkBookedStatus = new Text();
        Text bookStatusText = new Text();
        Text displayTotalCost = new Text();


        VBox buttonsForBooking = new VBox();
        buttonsForBooking.getChildren().addAll(availabilityCheck, bookOnClick);
        buttonsForBooking.setSpacing(5.0);

        allDates.getChildren().addAll(pickUpDateSec, dropOffDateSec, buttonsForBooking);

        bookingSection.getChildren().addAll(allDates, checkBookedStatus, bookStatusText,
displayTotalCost);

        bookingSection.setPadding(new Insets(20, 0, 20, 20));


        //-------------------------------------------------
```

```
        VBox parent = new VBox(allSearchFilter, tableOfVehicles, bookingSection);
        Scene newScene = new Scene(parent);
        primaryStage.setScene(newScene);
        primaryStage.show();
        primaryStage.setAlwaysOnTop(true);          //open the application on top of intelliJ

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

        //Button actions

        searchClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when search
button is clicked

            @Override
            public void handle(ActionEvent event) {

                String vehMakeSearch = makeSearch.getText();       //getting Make to be searched

                searchedVehicles.clear();          //clearing previous search results from ArrayList

                for (Vehicle searchVeh : WestminsterRentalVehicleManager.getVehiclesInSystem()) {
                    if (searchVeh.getMake().equals(vehMakeSearch)) {
                        searchedVehicles.add(searchVeh);      //adding vehicles that have matching makes
as searched into ArrayList
                    }
                }
//                System.out.println(searchedVehicles);      //to check

                tableOfVehicles.getItems().clear();     //clearing table
                tableOfVehicles.getItems().addAll(searchedVehicles);

            }
        });


        resetClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when reset
button is clicked

            @Override
            public void handle(ActionEvent event) {

                searchedVehicles.clear();       //resetting search to all Vehicles
                searchedVehicles.addAll(WestminsterRentalVehicleManager.getVehiclesInSystem());

                tableOfVehicles.getItems().clear();     //reseting display to all Vehicles

tableOfVehicles.getItems().addAll(WestminsterRentalVehicleManager.getVehiclesInSystem());
```

```
            makeSearch.setText("");
        }
    });


    filterCarClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when Filter
Cars button is clicked

        @Override
        public void handle(ActionEvent event) {

            searchInSearch.clear();

            for (Vehicle searchVeh : searchedVehicles) {
                if (searchVeh instanceof Car) {
                    searchInSearch.add(searchVeh);      //adding vehicles that are of Type Car into
ArrayList
                }
            }
//              System.out.println(searchInSearch);      //to check

            tableOfVehicles.getItems().clear();    //clearing table
            tableOfVehicles.getItems().addAll(searchInSearch);

        }
    });


    filterBikeClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when Filter
Motorbikes button is clicked

        @Override
        public void handle(ActionEvent event) {

            searchInSearch.clear();

            for (Vehicle searchVeh : searchedVehicles) {
                if (searchVeh instanceof Motorbike) {
                    searchInSearch.add(searchVeh);      //adding vehicles that are of Type Car into
ArrayList
                }
            }
//              System.out.println(searchInSearch);      //to check

            tableOfVehicles.getItems().clear();    //clearing table
            tableOfVehicles.getItems().addAll(searchInSearch);

        }
    });
```

```
    //-------------------------------------------------


    availabilityCheck.setOnAction(new EventHandler<ActionEvent>() {          //actions when
Availability check button is clicked

        @Override
        public void handle(ActionEvent event) {

          try {
            //getting input of pick up date
            Integer dayPickUpInput = Integer.parseInt(dayPickUp.getText());       //getting day
            Integer monthPickUpInput = Integer.parseInt(monthPickUp.getText());       //getting
month
            Integer yearPickUpInput = Integer.parseInt(yearPickUp.getText());       //getting year


            //getting input of drop off date
            Integer dayDropOffInput = Integer.parseInt(dayDropOff.getText());       //getting day
            Integer monthDropOffInput = Integer.parseInt(monthDropOff.getText());
//getting month
            Integer yearDropOffInput = Integer.parseInt(yearDropOff.getText());       //getting
year


            if (tableOfVehicles.getSelectionModel().getSelectedItem() != null) {

              Vehicle chosenVeh = (Vehicle)
tableOfVehicles.getSelectionModel().getSelectedItem();       //selected vehicle's information

//                System.out.println(chosenVeh);      //to check whether expected vehicle was
chosen

              Schedule newBooking = new Schedule(yearPickUpInput, monthPickUpInput,
dayPickUpInput,
                  yearDropOffInput, monthDropOffInput, dayDropOffInput);

              boolean availability = GUIController.checkAvailabilityOfVeh(chosenVeh,
newBooking);

              if (availability) {  //vehicle available
                checkBookedStatus.setFill(Color.GREEN);
//              System.out.println("Vehicle is available for booking.");
                checkBookedStatus.setText(chosenVeh.getPlateNo() + " is available for
booking.");

              } else {
```

```
                    checkBookedStatus.setFill(Color.RED);
//                  System.out.println("Vehicle isn't available for booking during requested time
period.");
                    checkBookedStatus.setText(chosenVeh.getPlateNo() + " isn't available for
booking during requested time period.");
                }
            } else {
                checkBookedStatus.setFill(Color.DARKGRAY);
                checkBookedStatus.setText("Please select a vehicle to book.");

            }
        } catch (NumberFormatException e) {
            checkBookedStatus.setFill(Color.DARKGRAY);
            checkBookedStatus.setText("Please enter a valid date in Integer Numbers.");
        }
    }
});


    bookOnClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when Book
button is clicked

        @Override
        public void handle(ActionEvent event) {

            try {

                //getting input of pick up date
                Integer dayPickUpInput = Integer.parseInt(dayPickUp.getText());       //getting day
                Integer monthPickUpInput = Integer.parseInt(monthPickUp.getText());       //getting
month
                Integer yearPickUpInput = Integer.parseInt(yearPickUp.getText());       //getting year


                //getting input of drop off date
                Integer dayDropOffInput = Integer.parseInt(dayDropOff.getText());       //getting day
                Integer monthDropOffInput = Integer.parseInt(monthDropOff.getText());
//getting month
                Integer yearDropOffInput = Integer.parseInt(yearDropOff.getText());       //getting
year


                if (tableOfVehicles.getSelectionModel().getSelectedItem() != null) {
                    //getting selected vehicle's information
                    Vehicle chosenVeh = (Vehicle)
tableOfVehicles.getSelectionModel().getSelectedItem();       //selected vehicle's information
                    //down-casted from Object type to Vehicle type
                    System.out.println(chosenVeh);      //to check whether expected vehicle was
chosen
```

```
                Schedule newBooking = new Schedule(yearPickUpInput, monthPickUpInput,
dayPickUpInput,
                        yearDropOffInput, monthDropOffInput, dayDropOffInput);

                boolean booked = GUIController.createBooking(chosenVeh, newBooking);

                if (booked) {

                    checkBookedStatus.setFill(Color.GREEN);
//                  System.out.println("Vehicle is available for booking.");
                    checkBookedStatus.setText(chosenVeh.getPlateNo() + " is available for
booking.");


                    bookStatusText.setText("Booked vehicle with Plate No: " +
chosenVeh.getPlateNo() + " from " +
                            newBooking.getPickUp() + " to " + newBooking.getDropOff());


                    //addToBookedDB here
                    DatabaseController.addToBookedDB(chosenVeh.getPlateNo(), yearPickUpInput,
monthPickUpInput, dayPickUpInput,
                            yearDropOffInput, monthDropOffInput, dayDropOffInput);


                    displayTotalCost.setText("Total Cost: $" +
GUIController.getCalculatedRent(chosenVeh.getDailyCost(), newBooking));
                } else {
                    //notify the user that the vehicle isn't available for rent during the chosen time
period.
                    checkBookedStatus.setFill(Color.RED);
//                  System.out.println("Vehicle isn't available for booking during requested time
period.");
                    checkBookedStatus.setText(chosenVeh.getPlateNo() + " isn't available for
booking during requested time period.");

                }
            } else {
                checkBookedStatus.setFill(Color.DARKGRAY);
                checkBookedStatus.setText("Please select a vehicle to book.");

            }
        } catch (NumberFormatException e) {
            checkBookedStatus.setFill(Color.DARKGRAY);
            checkBookedStatus.setText("Please enter a valid date in Integer Numbers.");
        }

    }
```

```
        });

    }
}
```

```
/*
References:

https://stackoverflow.com/questions/14169240/getting-integer-values-from-textfield

How to get information of selected row in javafx tableview
https://stackoverflow.com/questions/17388866/getting-selected-item-from-a-javafx-tableview
 */
```

GUIController

Code

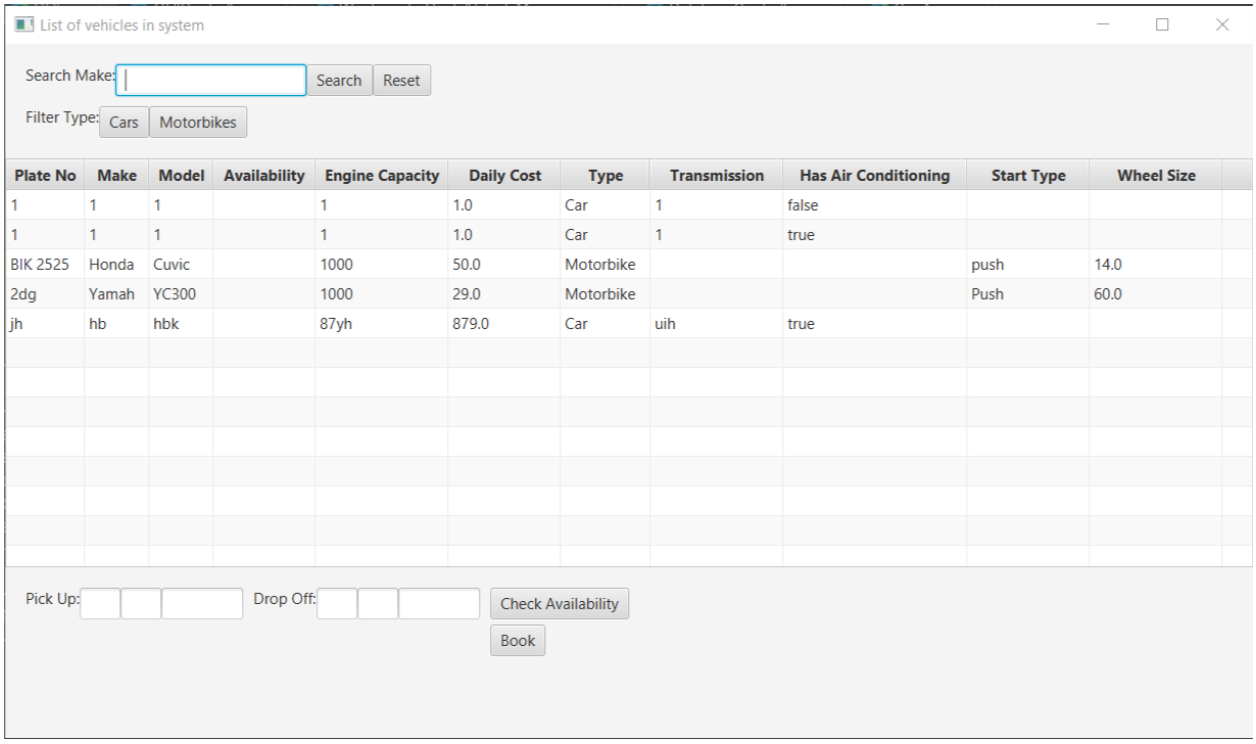
```
package lk.dinuka.VehicleRentalSystem.Controller;

import lk.dinuka.VehicleRentalSystem.Model.Schedule;
import lk.dinuka.VehicleRentalSystem.Model.Vehicle;

import java.math.BigDecimal;
import java.time.LocalDate;
import java.time.Period;
import java.util.ArrayList;
import java.util.List;

import static
lk.dinuka.VehicleRentalSystem.Controller.WestminsterRentalVehicleManager.bookedVehicles;


public class GUIController {

   public static boolean createBooking(Vehicle chosenVeh, Schedule newBooking ) {
      //used to create a booking as required and add booking info into the system

      List<Schedule> bookedVehicleDates = new ArrayList<>();     //used to record pick up & drop
off dates of a vehicle
      //Only used to store the dates into the bookedVehicles HashMap

      boolean availability = checkAvailabilityOfVeh(chosenVeh, newBooking);   //checking
whether vehicle is available for booking


      System.out.println();
      System.out.println("---checked availability---");
      System.out.println();

      if (availability) {
         System.out.println("Vehicle is available for booking");

         if (bookedVehicles.containsKey(chosenVeh.getPlateNo())) {
            bookedVehicleDates = bookedVehicles.get(chosenVeh.getPlateNo());          //getting
recorded bookings into temporary list
         }
         bookedVehicleDates.add(newBooking);     //adding the newly booked dates to the list of
bookings.

         WestminsterRentalVehicleManager.bookedVehicles.put(chosenVeh.getPlateNo(),
(ArrayList) bookedVehicleDates);      //adding all booked vehicles to bookedVehicles HashMap
```

```
        System.out.println(WestminsterRentalVehicleManager.bookedVehicles);        //checking
whether required booking was entered into the system
        return true;
      } else {
        System.out.println("Vehicle isn't available for booking during the requested time
period.");
        //vehicle isn't available to be book
        return false;
      }
  }


  //``````~~~~~~~~~~~~~~~~~~~~~~``````


  public static boolean checkAvailabilityOfVeh(Vehicle chosenVeh, Schedule newBooking ) {
    //used to check for the availability of a chosen vehicle

    String plateNoOfChosen = chosenVeh.getPlateNo();       //The plate number of the chosen
vehicle


    if (!WestminsterRentalVehicleManager.bookedVehicles.containsKey(plateNoOfChosen)) {
      return true;       //vehicle is not booked
    } else {

      List<Schedule> bookedVehicleDates = new ArrayList<>();    //used to record pick up &
drop off dates of a vehicle
      bookedVehicleDates = bookedVehicles.get(chosenVeh.getPlateNo());          //getting
recorded bookings into temporary list
      //Only used to get each of the dates from the bookedVehicles HashMap Values


      int totalBookings = bookedVehicles.get(plateNoOfChosen).size();
      int passedChecks = 0;

      for (int i = 0; i < totalBookings; i++) {

        boolean checkPickUpBefore = LocalDate.from(newBooking.getPickUp()).isBefore(
//pick up before booked pickup
            bookedVehicleDates.get(i).getPickUp());
        boolean checkDropOffBefore = LocalDate.from(newBooking.getDropOff()).isBefore(
//drop off before booked pick up
            bookedVehicleDates.get(i).getPickUp());

        boolean checkPickUpAfter = LocalDate.from(newBooking.getPickUp()).isAfter(
//pick up after booked drop off
            bookedVehicleDates.get(i).getDropOff());
```

```
            boolean checkDropOffAfter = LocalDate.from(newBooking.getDropOff()).isAfter(
//drop off after booked drop off
                    bookedVehicleDates.get(i).getDropOff());


            if ((checkPickUpBefore && checkDropOffBefore) || (checkPickUpAfter &&
checkDropOffAfter)) {
                // if both requested pick up and drop off are either before the booked pick up date
or after the
                // booked drop off date, the vehicle is available for requested period

                passedChecks += 1;

            }
            //if false for at least one, can't book
          }

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

//          if (totalBookings>0){
//              return passedChecks == totalBookings;          //if all the bookings don't interfere with
the requested time -> true
//
//          } else{
//              return true;
          //since this else block will run only if there has been at least one previous entry, the
above verification isn't required
          return passedChecks == totalBookings;          //if all the bookings don't interfere with the
requested time -> true

      }
   }

   public static BigDecimal getCalculatedRent(BigDecimal dailyCost, Schedule newBooking) {
//       have calculation of total cost here

      BigDecimal totalCost = BigDecimal.valueOf(0);

      Period period =
Period.between(newBooking.getPickUp(),newBooking.getDropOff());//difference between the
number of days
      int noOfDays = period.getDays();

      if (noOfDays > 0) {
         return dailyCost.multiply(BigDecimal.valueOf(noOfDays));    //dailyCost*noOfDays
      }
      return totalCost;
   }
}
```

Screenshots

Visualize the list of vehicles



Code

```
   @Override
   public void start(Stage primaryStage) throws Exception {
//       Platform.setImplicitExit(false);
       primaryStage.setTitle("List of vehicles in system");

       TableView tableOfVehicles = new TableView();

       //Creating columns to be added to the table
       TableColumn<String, Vehicle> plateNoColumn = new TableColumn<>("Plate No");
       plateNoColumn.setCellValueFactory(new PropertyValueFactory<>("plateNo"));

       TableColumn<String, Vehicle> makeColumn = new TableColumn<>("Make");
       makeColumn.setCellValueFactory(new PropertyValueFactory<>("make"));

       TableColumn<String, Vehicle> modelColumn = new TableColumn<>("Model");
       modelColumn.setCellValueFactory(new PropertyValueFactory<>("model"));

       TableColumn<String, Vehicle> availabilityColumn = new TableColumn<>("Availability");
       availabilityColumn.setCellValueFactory(new PropertyValueFactory<>("availability"));
       availabilityColumn.setMinWidth(100);

       TableColumn<String, Vehicle> engineCapacityColumn = new TableColumn<>("Engine
Capacity");
```

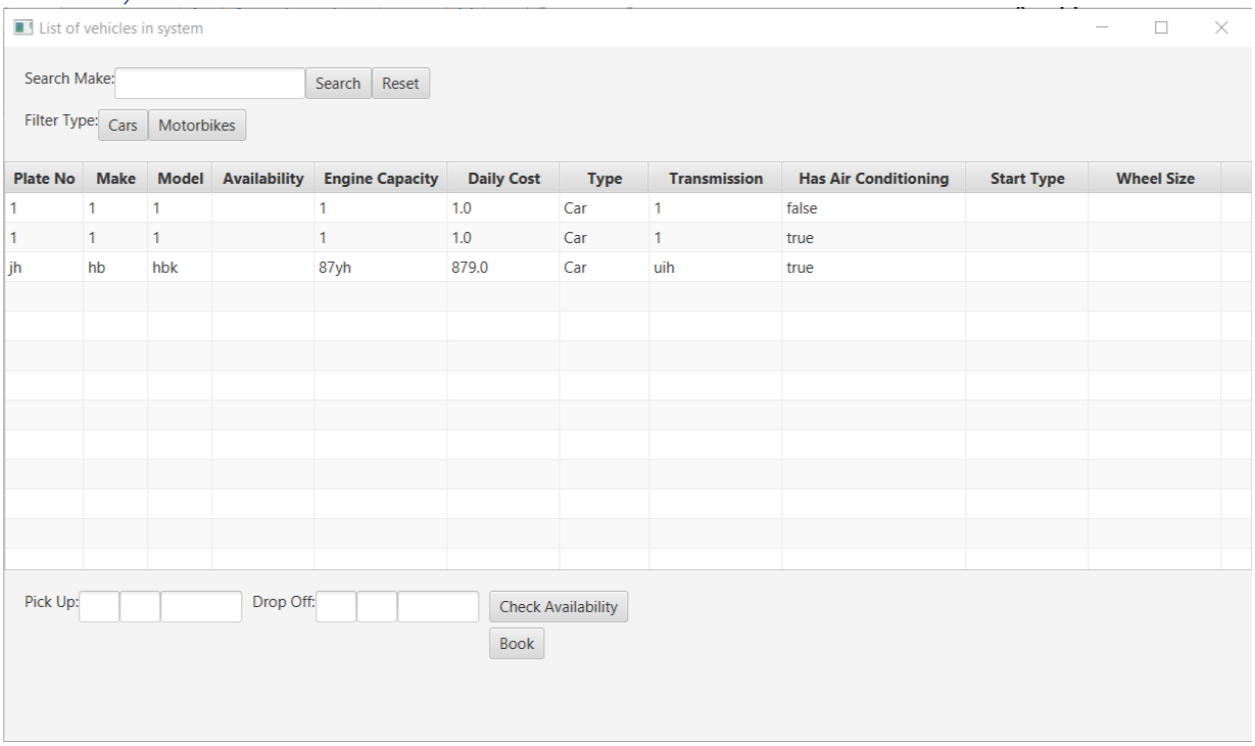
```
        engineCapacityColumn.setCellValueFactory(new
PropertyValueFactory<>("engineCapacity"));
        engineCapacityColumn.setMinWidth(130);

        TableColumn<String, Vehicle> dailyCostColumn = new TableColumn<>("Daily Cost");
        dailyCostColumn.setCellValueFactory(new PropertyValueFactory<>("dailyCost"));
        dailyCostColumn.setMinWidth(110);

        TableColumn<String, Vehicle> typeColumn = new TableColumn<>("Type");
        typeColumn.setCellValueFactory(new PropertyValueFactory<>("type"));

        TableColumn<String, Vehicle> transmissionColumn = new TableColumn<>("Transmission");
        transmissionColumn.setCellValueFactory(new PropertyValueFactory<>("transmission"));
        transmissionColumn.setMinWidth(130);

        TableColumn<String, Vehicle> hasAirConColumn = new TableColumn<>("Has Air
Conditioning");
        hasAirConColumn.setCellValueFactory(new PropertyValueFactory<>("hasAirCon"));
        hasAirConColumn.setMinWidth(180);

        TableColumn<String, Vehicle> startTypeColumn = new TableColumn<>("Start Type");
        startTypeColumn.setCellValueFactory(new PropertyValueFactory<>("startType"));
        startTypeColumn.setMinWidth(120);

        TableColumn<String, Vehicle> wheelSizeColumn = new TableColumn<>("Wheel Size");
        wheelSizeColumn.setCellValueFactory(new PropertyValueFactory<>("wheelSize"));
        wheelSizeColumn.setMinWidth(130);


        tableOfVehicles.getColumns().addAll(plateNoColumn, makeColumn, modelColumn,
availabilityColumn, engineCapacityColumn, dailyCostColumn,
                typeColumn, transmissionColumn, hasAirConColumn, startTypeColumn,
wheelSizeColumn);          //adding all the columns to the table


tableOfVehicles.getItems().addAll(WestminsterRentalVehicleManager.getVehiclesInSystem());
//adding all the vehicles in the available
        // in the vehiclesInSystem ArrayList


        searchedVehicles.addAll(WestminsterRentalVehicleManager.getVehiclesInSystem());
//to get filter by vehicle type to work before searching for a Make
```

Explanation of the functionality

When the GUI is opened, the list of all vehicles is displayed as shown above.

Filter vehicles by type

Filtered by cars



Code

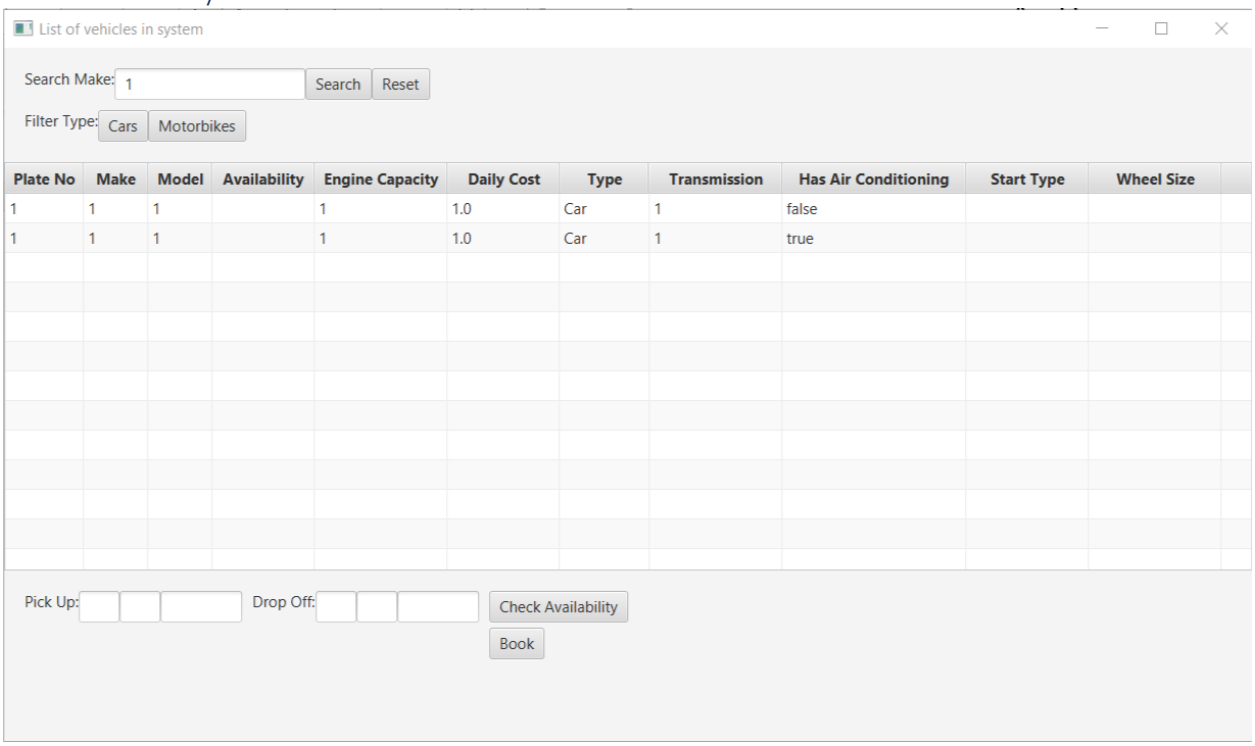
```
filterCarClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when Filter Cars button is clicked

        @Override
        public void handle(ActionEvent event) {

          searchInSearch.clear();

          for (Vehicle searchVeh : searchedVehicles) {
            if (searchVeh instanceof Car) {
              searchInSearch.add(searchVeh);      //adding vehicles that are of Type Car into ArrayList
            }
          }
//          System.out.println(searchInSearch);      //to check

          tableOfVehicles.getItems().clear();    //clearing table
          tableOfVehicles.getItems().addAll(searchInSearch);

        }
    });


    filterBikeClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when Filter Motorbikes button is clicked
```

```
        @Override
        public void handle(ActionEvent event) {

            searchInSearch.clear();

            for (Vehicle searchVeh : searchedVehicles) {
                if (searchVeh instanceof Motorbike) {
                    searchInSearch.add(searchVeh);      //adding vehicles that are of Type Car into
ArrayList
                }
            }
//              System.out.println(searchInSearch);      //to check

            tableOfVehicles.getItems().clear();    //clearing table
            tableOfVehicles.getItems().addAll(searchInSearch);

        }
    });
```

Explanation of the functionality

When the "Filter Type: Cars" button is clicked, all the cars in the system are displayed in the table. Similarly, it works for Motorbikes as well.

Filter vehicles by make



Code

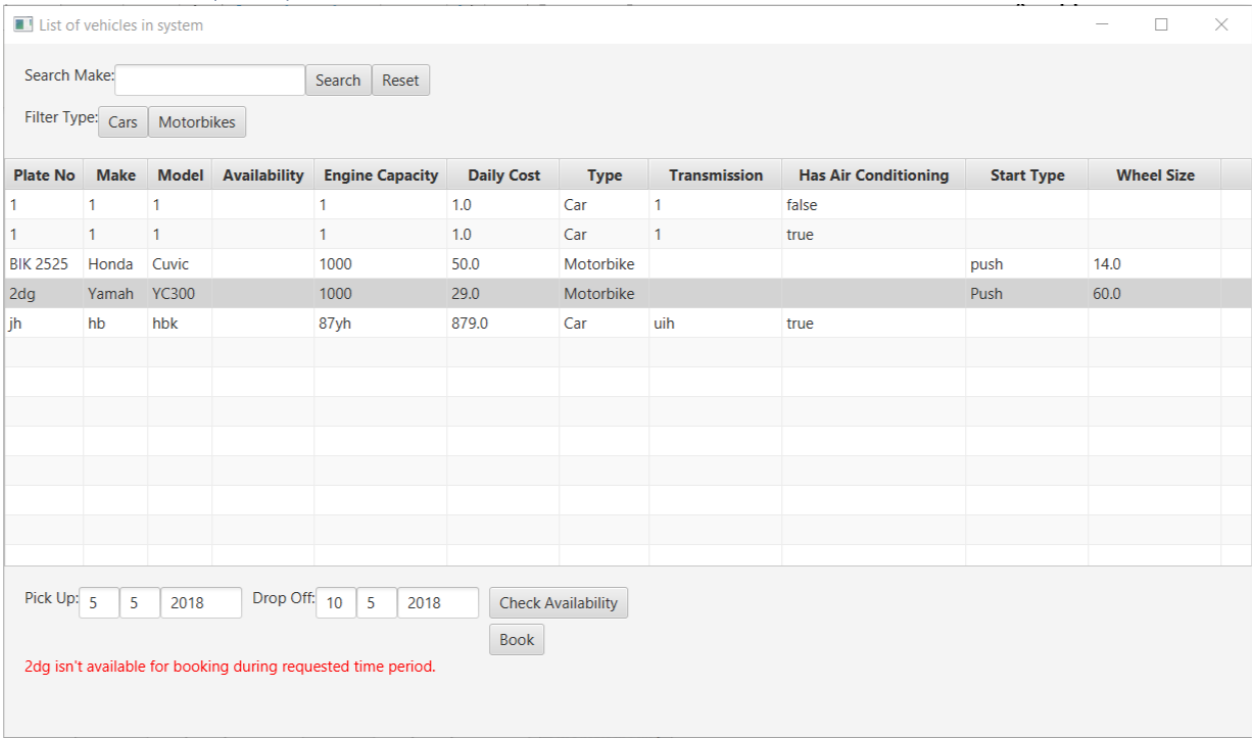
```
    searchClick.setOnAction(new EventHandler<ActionEvent>() {          //actions when search
button is clicked

        @Override
        public void handle(ActionEvent event) {
```

```
            String vehMakeSearch = makeSearch.getText();       //getting Make to be searched

            searchedVehicles.clear();         //clearing previous search results from ArrayList

            for (Vehicle searchVeh : WestminsterRentalVehicleManager.getVehiclesInSystem()) {
               if (searchVeh.getMake().equals(vehMakeSearch)) {
                  searchedVehicles.add(searchVeh);      //adding vehicles that have matching makes
as searched into ArrayList
               }
            }
//             System.out.println(searchedVehicles);      //to check

            tableOfVehicles.getItems().clear();    //clearing table
            tableOfVehicles.getItems().addAll(searchedVehicles);

         }
      });
```

Explanation of the functionality

When the required make is typed in and search using the search box, all vehicles related to the searched Make is displayed.
When researched is clicked all Filters and search results are removed, displaying all the vehicles available in the system.

Check availability on specific dates



Code

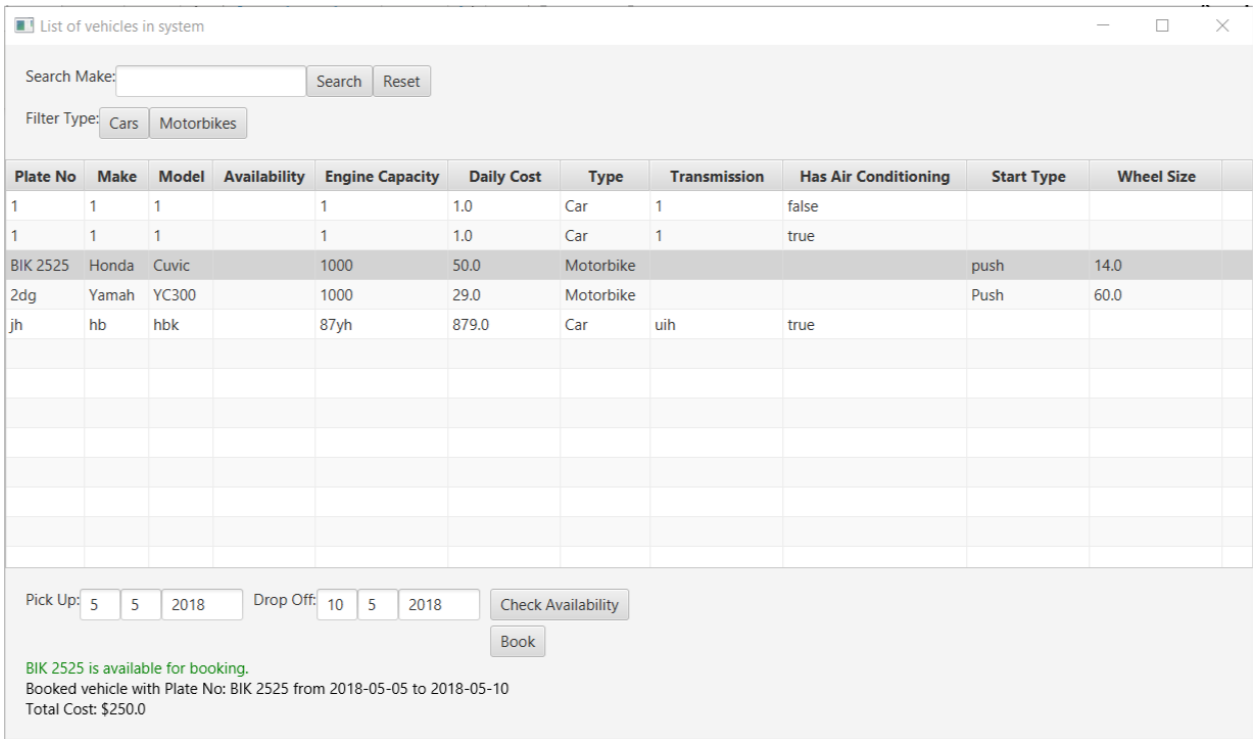
```
availabilityCheck.setOnAction(new EventHandler<ActionEvent>() {          //actions when
Availability check button is clicked

        @Override
        public void handle(ActionEvent event) {

          try {
            //getting input of pick up date
            Integer dayPickUpInput = Integer.parseInt(dayPickUp.getText());       //getting day
            Integer monthPickUpInput = Integer.parseInt(monthPickUp.getText());       //getting
month
            Integer yearPickUpInput = Integer.parseInt(yearPickUp.getText());       //getting year


            //getting input of drop off date
            Integer dayDropOffInput = Integer.parseInt(dayDropOff.getText());       //getting day
            Integer monthDropOffInput = Integer.parseInt(monthDropOff.getText());
//getting month
            Integer yearDropOffInput = Integer.parseInt(yearDropOff.getText());       //getting
year


            if (tableOfVehicles.getSelectionModel().getSelectedItem() != null) {

              Vehicle chosenVeh = (Vehicle)
tableOfVehicles.getSelectionModel().getSelectedItem();       //selected vehicle's information

//                System.out.println(chosenVeh);      //to check whether expected vehicle was
chosen

              Schedule newBooking = new Schedule(yearPickUpInput, monthPickUpInput,
dayPickUpInput,
                  yearDropOffInput, monthDropOffInput, dayDropOffInput);

              boolean availability = GUIController.checkAvailabilityOfVeh(chosenVeh,
newBooking);

              if (availability) {  //vehicle available
                checkBookedStatus.setFill(Color.GREEN);
//              System.out.println("Vehicle is available for booking.");
                checkBookedStatus.setText(chosenVeh.getPlateNo() + " is available for
booking.");

              } else {
                checkBookedStatus.setFill(Color.RED);
//              System.out.println("Vehicle isn't available for booking during requested time
period.");
```

```
                checkBookedStatus.setText(chosenVeh.getPlateNo() + " isn't available for
booking during requested time period.");
                }
            } else {
                checkBookedStatus.setFill(Color.DARKGRAY);
                checkBookedStatus.setText("Please select a vehicle to book.");

            }
        } catch (NumberFormatException e) {
            checkBookedStatus.setFill(Color.DARKGRAY);
            checkBookedStatus.setText("Please enter a valid date in Integer Numbers.");
        }
    }
});
```

Explanation of the functionality

When “Check Availability” is clicked after a vehicle is selected and pick up & drop off dates are specified, the system will let the user know whether the chosen vehicles is available or not during the requested time period.

Book vehicle



Code

```
    bookOnClick.setOnAction(new EventHandler<ActionEvent>() {        //actions when Book
button is clicked

        @Override
        public void handle(ActionEvent event) {
```

```
            try {

                //getting input of pick up date
                Integer dayPickUpInput = Integer.parseInt(dayPickUp.getText());       //getting day
                Integer monthPickUpInput = Integer.parseInt(monthPickUp.getText());       //getting
month
                Integer yearPickUpInput = Integer.parseInt(yearPickUp.getText());       //getting year


                //getting input of drop off date
                Integer dayDropOffInput = Integer.parseInt(dayDropOff.getText());       //getting day
                Integer monthDropOffInput = Integer.parseInt(monthDropOff.getText());
//getting month
                Integer yearDropOffInput = Integer.parseInt(yearDropOff.getText());       //getting
year


                if (tableOfVehicles.getSelectionModel().getSelectedItem() != null) {
                    //getting selected vehicle's information
                    Vehicle chosenVeh = (Vehicle)
tableOfVehicles.getSelectionModel().getSelectedItem();       //selected vehicle's information
                    //down-casted from Object type to Vehicle type
                    System.out.println(chosenVeh);      //to check whether expected vehicle was
chosen

                    Schedule newBooking = new Schedule(yearPickUpInput, monthPickUpInput,
dayPickUpInput,
                            yearDropOffInput, monthDropOffInput, dayDropOffInput);

                    boolean booked = GUIController.createBooking(chosenVeh, newBooking);

                    if (booked) {

                        checkBookedStatus.setFill(Color.GREEN);
//                      System.out.println("Vehicle is available for booking.");
                        checkBookedStatus.setText(chosenVeh.getPlateNo() + " is available for
booking.");


                        bookStatusText.setText("Booked vehicle with Plate No: " +
chosenVeh.getPlateNo() + " from " +
                                newBooking.getPickUp() + " to " + newBooking.getDropOff());


                        //addToBookedDB here
                        DatabaseController.addToBookedDB(chosenVeh.getPlateNo(), yearPickUpInput,
monthPickUpInput, dayPickUpInput,
                                yearDropOffInput, monthDropOffInput, dayDropOffInput);
```

```
                    displayTotalCost.setText("Total Cost: $" +
GUIController.getCalculatedRent(chosenVeh.getDailyCost(), newBooking));
                } else {
                    //notify the user that the vehicle isn't available for rent during the chosen time
period.
                    checkBookedStatus.setFill(Color.RED);
//                  System.out.println("Vehicle isn't available for booking during requested time
period.");
                    checkBookedStatus.setText(chosenVeh.getPlateNo() + " isn't available for
booking during requested time period.");

                }
            } else {
                checkBookedStatus.setFill(Color.DARKGRAY);
                checkBookedStatus.setText("Please select a vehicle to book.");

            }
        } catch (NumberFormatException e) {
            checkBookedStatus.setFill(Color.DARKGRAY);
            checkBookedStatus.setText("Please enter a valid date in Integer Numbers.");
        }

    }
});
```

Explanation of the functionality

When 'Book' button is clicked, the system will perform a similar check like "Check Availability" and let the user know that the vehicle was booked for the requested time period.
The total cost will also be displayed below.