UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

Department of Computing

(B.Eng.) in Software Engineering

Module: 5COSC007C.1: OOP

Module Leader: Mr. Guhanathan Poravi

Course Work

Dhanasekara Mudiyanselage Akshaan Dileesha Bandara

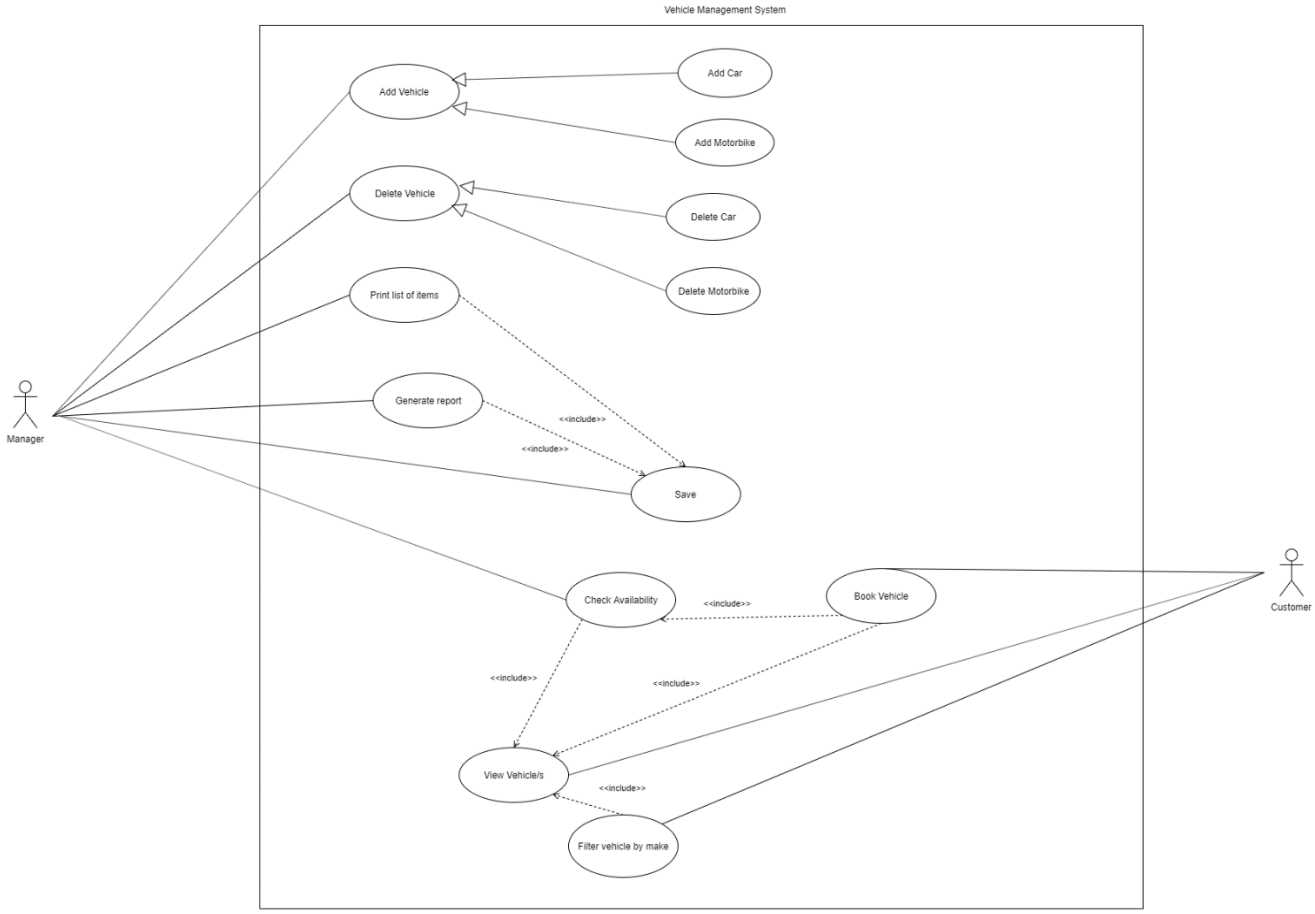
w1743055

2018597

Contents

Design

1. Use Case Diagram



2. Use Case description

Use Case No.: 1
Priority: High

Use Case name: Add Vehicle

Actor:

1) **Primary Actor:** Manager

Triggering Event: addVehicle_option_selected

Pre-Condition: hasSpace = true

Main Sequence:

Manager	System
2) Enter type of Vehicle 5) Enter details	1) Prompt for type of Vehicle 3) Validates type of vehicle 4) Prompt for details 6) Validates details 7) Show message "Addition successful" 8) Add Vehicle

Alternative Scenario 1:

Manager	System
6) Reenter	4) Show error 5) Prompt for reenter

Alternative Scenario 2:

Manager	System
	7) Show error 8) Go to 1)

Exceptional Scenario: Connection to database fails

Post Condition: successful: isAddSuccessFull = true

unsuccessful: isAddSuccessFull = false

Inclusions: None

Extensions: None

Use Case No.: 2
Priority: High

Use Case name: Delete Vehicle

Actor:

1) **Primary Actor:** Manager

Triggering Event: deleteVehicle_option_selected

Pre-Condition: None

Main Sequence:

Manager	System
2) Enter plate number	1) Prompt for plate number 3) Validates plate number 4) Show message "Delete successful" 5) Delete Vehicle

Alternative Scenario 1:

Manager	System
	4) Show message "Vehicle not found" 8) Exit process

Exceptional Scenario: Connection to database fails

Post Condition: successful: isDeleteSuccessFull = true

unsuccessful: isDeleteSuccessFull = false

Inclusions: None

Extensions: None

Use Case No.: 3
Priority: High

Use Case name: Print list of Vehicles

Actor:

1) **Primary Actor:** Manager

Triggering Event: printAllVehicles_option_selected

Pre-Condition: noOfVehicles>0

Main Sequence:

Manager	System
1) Print all Vehicles option selected	2) Print all the recently added Vehicles 3) Print all the existing Vehicles

Alternative Scenario: None

Exceptional Scenario: Connection to database fails

Post Condition: None

Inclusions: None

Extensions: None

Use Case No.: 4
Priority: High

Use Case name: Save Changes

Actor:

1) **Primary Actor:** Manager

Triggering Event: saveChanges_option_selected

Pre-Condition: None

Main Sequence:

Manager	System
1) Selects Save Changes option	2) Save the recent changes 3) Show message “Saved Successful”

Exceptional Scenario: Show message “File upload crashed”

Post Condition: successful: isSaveChanges = true

unsuccessful: isSaveChanges = false

Inclusions: None

Extensions: None

Use Case No.: 5
Priority: High

Use Case name: Generate Report

Actor:

1) **Primary Actor:** Manager

Triggering Event: generateReport_option_selected

Pre-Condition: None

Main Sequence:

Manager	System
1) Selects generate report option	2) Show all details of recently changes in the file

Alternative Scenario: None

Exceptional Scenario: Connectivity between file and the system fails or Connection to database fails

Post Condition: None

Inclusions: Save Changes

Extensions: None

Use Case No.: 6
Priority: High

Use Case name: Check Availability

Actor:

1) **Primary Actor:** Manager

Triggering Event: isCheckAvailability_option_selected

Pre-Condition: plateNumber

Main Sequence:

Manager	System
1) Enters the plate number	2) Search for the Vehicle with the entered plate number 3) Show details of the vehicle

Alternative Scenario 1:

Manager	System
	3) Show message nothing if no vehicle exists 4) Exit process

Exceptional Scenario: Connectivity between back end and front end fails

Post Condition: successful: isFoundVehicle = true

unsuccessful: isFoundVehicle = false

Inclusions: View Vehicle/s

Extensions: None

Use Case No.: 7
Priority: High

Use Case name: View Vehicles

Actor:

1) **Primary Actor:** Customer

Triggering Event: View_all_Vehicle_option_clicked

Pre-Condition: None

Main Sequence:

Customer	System
1) Clicks the view all vehicles option	2) Prompt for Viewing Cars or Motorbikes
3) Clicks Cars or Motorbikes	4) Display the Vehicle/s with details

Alternative Scenario: None

Exceptional Scenario: Connection between front end and back end fails

Post Condition: successful: display the list of vehicles

unsuccessful: no printing list of vehicles

Inclusions: None

Extensions: None

Use Case No.: 8
Priority: High

Use Case name: Filter Vehicles by make

Actor:

1) **Primary Actor:** Customer

Triggering Event: typed_text

Pre-Condition: None

Main Sequence:

Customer	System
1) Type some name of a make	2) Filters vehicles according to the make 3) Print the details of Vehicle

Alternative Scenario 1:

Customer	System
	2) No vehicle found of name of the text that user typed 3) Not showing any vehicle

Exceptional Scenario: Connection between front end and back end failed

Post Condition: successful: Print details of Vehicles according to the filtering by make

unsuccessful: Show no vehicle

Inclusions: View Vehicles

Extensions: None

Use Case No.: 9
Priority: High

Use Case name: Book Vehicle

Actor:

1) **Primary Actor:** Customer

Triggering Event: bookVehicle_button_clicked

Pre-Condition: All details entered

Main Sequence:

Customer	System
1) Selects Vehicle to wished to be book 2) Enter the customer details	3) Validates the details that is been entered 4) Book the vehicle 5) Show “Booking Successful” message 6) Show Thankyou message

Alternative Scenario 1:

Customer	System
6) Reenter	4) Show error 5) Prompt for reenter

Exceptional Scenario: Connection between front end and back end fails

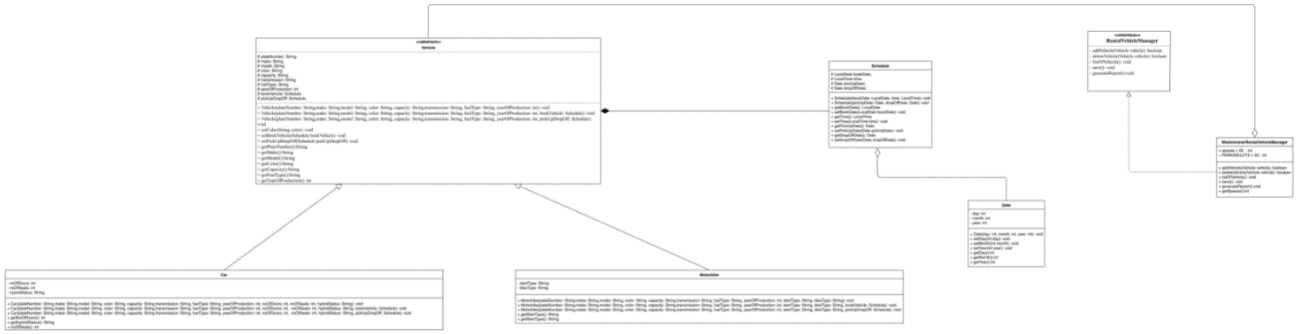
Post Condition: successful: isBookingSuccessFull = true

unsuccessful: isBookingSuccessFull = false

Inclusions: View Vehicle/s, Check availability

Extensions: None

3. Class Diagram



Development

Technology Stack

Front end: Angular

Back end: Spring boot

Database: MongoDB

Code

OOP

Vehicle.java

```
package CW;
import CW.Schedule;

import java.util.Objects;


public abstract class Vehicle implements Comparable<Vehicle>{
    protected String plateNumber;
    protected String make;
    protected String model;
    protected String color;
    protected String capacity;
    protected String transmission;
    protected String fuelType;
    protected int yearOfProduction;
    protected Schedule bookVehicle;
    protected Schedule pickUpDropOff;


    public Vehicle(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction) {
        super();
        this.plateNumber = plateNumber;
        this.make = make;
        this.model = model;
        this.color = color;
        this.capacity = capacity;
        this.transmission = transmission;
        this.fuelType = fuelType;
        this.yearOfProduction = yearOfProduction;
    }

    public Vehicle(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction, Schedule
bookVehicle) {
```

```
        super();
        this.plateNumber = plateNumber;
        this.make = make;
        this.model = model;
        this.color = color;
        this.capacity = capacity;
        this.transmission = transmission;
        this.fuelType = fuelType;
        this.yearOfProduction = yearOfProduction;
        this.bookVehicle = bookVehicle;
    }

    public Vehicle(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, Schedule pickUpDropOff ,int
yearOfProduction) {
        super();
        this.plateNumber = plateNumber;
        this.make = make;
        this.model = model;
        this.color = color;
        this.capacity = capacity;
        this.transmission = transmission;
        this.fuelType = fuelType;
        this.yearOfProduction = yearOfProduction;
        this.pickUpDropOff = pickUpDropOff;
    }

    //setters only for color, bookVehicle and pickUpDropOff, because states other
than color, bookVehicle and pickUpDropOff are unique for vehicle

    public String getPlateNumber() {
        return plateNumber;
    }


    public String getMake() {
        return make;
    }


    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }



    public String getCapacity() {
        return capacity;
    }
```

```
public String getTransmission() {
    return transmission;
}


public String getFuelType() {
    return fuelType;
}


public int getYearOfProduction() {
    return yearOfProduction;
}


public Schedule getBookVehicle() {
    return bookVehicle;
}

public void setBookVehicle(Schedule bookVehicle) {
    this.bookVehicle = bookVehicle;
}

public Schedule getPickUpDropOff() {
    return pickUpDropOff;
}

public void setPickUpDropOff(Schedule pickUpDropOff) {
    this.pickUpDropOff = pickUpDropOff;
}

public String getModel() {
    return model;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Vehicle vehicle = (Vehicle) o;
    return yearOfProduction == vehicle.yearOfProduction &&
            plateNumber.equals(vehicle.plateNumber) &&
            make.equals(vehicle.make) &&
            model.equals(vehicle.model) &&
            color.equals(vehicle.color) &&
            capacity.equals(vehicle.capacity) &&
            transmission.equals(vehicle.transmission) &&
            fuelType.equals(vehicle.fuelType) &&
            bookVehicle.equals(vehicle.bookVehicle) &&
            pickUpDropOff.equals(vehicle.pickUpDropOff);
}

@Override
public int hashCode() {
    return Objects.hash(plateNumber, make, model, color, capacity, transmission,
```

```
fuelType, yearOfProduction, bookVehicle, pickUpDropOff);
    }

    @Override
    public String toString() {
        return "Vehicle{" +
                "plateNumber='" + plateNumber + '\'' +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                ", capacity='" + capacity + '\'' +
                ", transmission='" + transmission + '\'' +
                ", fuelType='" + fuelType + '\'' +
                ", yearOfProduction=" + yearOfProduction +
                ", bookVehicle=" + bookVehicle +
                ", pickUpDropOff=" + pickUpDropOff +
                '}';
    }

    @Override
    public int compareTo(Vehicle o) {
        return this.getMake().compareTo(o.getMake());
    }
}
```

Car.java

```
package CW;

import java.util.Objects;
public class Car extends Vehicle {
    private int noOfDoors;
    private int noOfSeats;
    private String hybridStatus;

    public Car(String plateNumber, String make, String model, String color, String
capacity, String transmission, String fuelType, int yearOfProduction, int noOfDoors,
int noOfSeats,String hybridStatus) {
        super(plateNumber, make, model,color, capacity, transmission, fuelType,
yearOfProduction);
        this.noOfDoors = noOfDoors;
        this.noOfSeats = noOfSeats;
        this.hybridStatus = hybridStatus;
    }

    public Car(String plateNumber, String make, String model, String color, String
capacity, String transmission, String fuelType, int yearOfProduction, Schedule
bookVehicle, int noOfDoors,int noOfSeats, String hybridStatus) {
        super(plateNumber, make, model, color, capacity, transmission, fuelType,
yearOfProduction, bookVehicle);
        this.noOfDoors = noOfDoors;
        this.noOfSeats = noOfSeats;
        this.hybridStatus = hybridStatus;
    }
```

```
    public Car(String plateNumber, String make, String model, String color, String
capacity, String transmission, String fuelType, Schedule pickUpDropOff, int
yearOfProduction, int noOfDoors,int noOfSeats, String hybridStatus) {
        super(plateNumber, make, model, color, capacity, transmission, fuelType,
pickUpDropOff, yearOfProduction);
        this.noOfDoors = noOfDoors;
        this.noOfSeats = noOfSeats;
        this.hybridStatus = hybridStatus;
    }

    //no setters because all the states are unique for car

    public int getNoOfDoors() {
        return noOfDoors;
    }


    public String getHybridStatus() {
        return hybridStatus;
    }

    public int getNoOfSeats() {
        return noOfSeats;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        Car car = (Car) o;
        return noOfDoors == car.noOfDoors &&
                noOfSeats == car.noOfSeats &&
                hybridStatus.equals(car.hybridStatus);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), noOfDoors, noOfSeats, hybridStatus);
    }

    @Override
    public String toString() {
        return "Car{" +
                "plateNumber='" + plateNumber + '\'' +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                ", capacity='" + capacity + '\'' +
                ", transmission='" + transmission + '\'' +
                ", fuelType='" + fuelType + '\'' +
                ", yearOfProduction=" + yearOfProduction +
                ", bookVehicle=" + bookVehicle +
                ", pickUpDropOff=" + pickUpDropOff +
```

```
                ", noOfDoors=" + noOfDoors +
                ", noOfSeats=" + noOfSeats +
                ", hybridStatus='" + hybridStatus +
                '}';
    }
}
```

Motorbike.java

```
package CW;

import java.util.Objects;
public class Motorbike extends Vehicle {
    private String startType;
    private String bikeType;

    public Motorbike(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction, String
startType, String bikeType) {
        super(plateNumber, make, model, color, capacity, transmission, fuelType,
yearOfProduction);
        this.startType = startType;
        this.bikeType = bikeType;
    }

    public Motorbike(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction, Schedule
bookVehicle, String startType, String bikeType) {
        super(plateNumber, make, model, color, capacity, transmission, fuelType,
yearOfProduction, bookVehicle);
        this.startType = startType;
        this.bikeType = bikeType;
    }

    public Motorbike(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, Schedule pickUpDropOff, int
yearOfProduction, String startType, String bikeType) {
        super(plateNumber, make, model, color, capacity, transmission, fuelType,
pickUpDropOff, yearOfProduction);
        this.startType = startType;
        this.bikeType = bikeType;
    }

    //no setters because all the states are unique for motorbike

    public String getStartType() {
        return startType;
    }

    public String getBikeType() {
        return bikeType;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        Motorbike motorbike = (Motorbike) o;
        return startType.equals(motorbike.startType) &&
                bikeType.equals(motorbike.bikeType);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), startType, bikeType);
    }

    @Override
    public String toString() {
        return "Motorbike{" +
                "plateNumber='" + plateNumber + '\'' +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                ", capacity='" + capacity + '\'' +
                ", transmission='" + transmission + '\'' +
                ", fuelType='" + fuelType + '\'' +
                ", yearOfProduction=" + yearOfProduction +
                ", bookVehicle=" + bookVehicle +
                ", pickUpDropOff=" + pickUpDropOff +
                ", startType='" + startType +
                ", bikeType='" + bikeType +
                '}';
    }
}
```

Schedule.java

```
package CW;

import java.time.LocalDate;
import java.time.LocalTime;

public class Schedule {
    protected LocalDate bookDate;
    protected LocalTime time;
    protected Date pickUpDate;
    protected Date dropOffDate;

    public Schedule(LocalDate bookDate, LocalTime time) {
        this.bookDate = bookDate;
        this.time = time;
    }
```

```
    public Schedule(Date pickUpDate, Date dropOffDate) {
        this.pickUpDate = pickUpDate;
        this.dropOffDate = dropOffDate;
    }

    public LocalDate getBookDate() {
        return bookDate;
    }

    public void setBookDate(LocalDate bookDate) {
        this.bookDate = bookDate;
    }

    public LocalTime getTime() {
        return time;
    }

    public void setTime(LocalTime time) {
        this.time = time;
    }

    public Date getPickUpDate() {
        return pickUpDate;
    }

    public void setPickUpDate(Date pickUpDate) {
        this.pickUpDate = pickUpDate;
    }

    public Date getDropOffDate() {
        return dropOffDate;
    }

    public void setDropOffDate(Date dropOffDate) {
        this.dropOffDate = dropOffDate;
    }
}
```

Date.java

```
package CW;

import java.util.Objects;

public class Date {
    private int day;
    private int month;
    private int year;

    public Date(int day, int month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }
```

```
    public int getDay() {
        return day;
    }

    public void setDay(int day) {
        this.day = day;
    }

    public int getMonth() {
        return month;
    }

    public void setMonth(int month) {
        this.month = month;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Date date = (Date) o;
        return day == date.day &&
                month == date.month &&
                year == date.year;
    }

    @Override
    public int hashCode() {
        return Objects.hash(day, month, year);
    }

    @Override
    public String toString() {
        return "Date{" +
                "day=" + day +
                ", month=" + month +
                ", year=" + year +
                '}';
    }
}
```

RentalVehicleManager.java

```
package CW;

public interface RentalVehicleManager {
    boolean addVehicle(Vehicle vehicle); //the user can add vehicles to the system
    boolean deleteVehicle(Vehicle vehicle);//the user can delete vehicles to the
system
    boolean listOfVehicle(); //the user can print the list of vehicles in the system
    boolean save(); //the user can save the changes to the system
    void generateReport(); //the user can generate a report of the recent changes
}
```

WestminsterRentalVehicleManager.java

```
package CW;

import java.awt.*;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

public class WestminsterRentalVehicleManager implements RentalVehicleManager{
    final int PARKINGLOTS = 50; //there are exactly 50 spaces for vehicles
    public static int spaces = 50; //spaces is the current no of spaces remain
initially starting from 50
    public static List<Vehicle> vehicles = new ArrayList<>(); //vehicles that are
added contains in this list
    public static List<Vehicle> deletedVehicles = new ArrayList<>(); //vehicles that
are deleted contains in this list
    private static File file = new File("Vehicles.txt"); //inside paranthesis is the
name of the file

    @Override
    public boolean addVehicle(Vehicle vehicle){
        if (vehicles.size()<PARKINGLOTS) {
            vehicles.add(vehicle);
            spaces-=1;
            System.out.println(vehicle+" addition successful!");
            System.out.println("No of spaces used: "+getSpaces());
        }else {
            System.out.println("No space!");
        }
        return true;
    }

    @Override
    public boolean deleteVehicle(Vehicle vehicle) {
        deletedVehicles.add(vehicle);
        vehicles.remove(vehicle);
        spaces+=1;
        System.out.println(vehicle+" deletion successful!");
        System.out.println("No of available spaces: "+getSpaces());
```

```
        return true;
    }

    @Override
    public boolean listOfVehicle() {
        System.out.println(getVehicles());
        return true;
    }

    @Override
    public boolean save() {
        try {
            FileWriter fileWriter = new FileWriter(file);
            BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
            bufferedWriter.write("");
            bufferedWriter.newLine();
            bufferedWriter.write("-------------------------------------------------
---------------Vehicles-------------------------------------------------------
----");
            bufferedWriter.newLine();
            bufferedWriter.write("");
            bufferedWriter.newLine();
            bufferedWriter.write("");
            bufferedWriter.newLine();
            for (int i = 0; i < getVehicles().size(); i++) {
                bufferedWriter.write((i+1)+") "+getVehicles().get(i));
                bufferedWriter.newLine();
                bufferedWriter.write("");
                bufferedWriter.newLine();
            }
            bufferedWriter.close();
        }catch (Exception e){
            System.out.println("File upload crashed!");
        }
        System.out.println("Saved successfully!");
        return true;
    }

    @Override
    public void generateReport() {
        try {
            if(getVehicles().size()==0){
                System.out.println("No vehicles!. Therefore, none to be shown...");
            }else {
                Desktop desktop = Desktop.getDesktop();
                if (!Desktop.isDesktopSupported()) {
                    System.out.println("Not Supported!");
                }
                if (file.exists()) {
                    desktop.open(file);
                } else {
                    System.out.println("File cannot be found!");
                }
            }
        }catch (Exception e){
```

```
            System.out.println("File opening failed!");
        }
    }

    public static int getSpaces() {
        return spaces;
    }


    public List<Vehicle> getVehicles() {
        return vehicles;
    }

    public static void setVehicles(List<Vehicle> vehicles) {
        WestminsterRentalVehicleManager.vehicles = vehicles;
    }


    public static List<Vehicle> getRDeletedVehicles() {
        return deletedVehicles;
    }

    public static void setDeletedVehicles(List<Vehicle> rentedVehicles) {
        WestminsterRentalVehicleManager.deletedVehicles = rentedVehicles;
    }
}

class MakeComparator implements Comparator<Vehicle>{ //compares Vehicles according to
the make
    @Override
    public int compare(Vehicle o1, Vehicle o2) {
        return o1.getMake().compareTo(o2.getMake());
    }
}
```

Manager.java

```
package CW;

import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import org.bson.Document;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

```
public class Manager {
    private static WestminsterRentalVehicleManager manager = new
WestminsterRentalVehicleManager();
    private static Scanner input = new Scanner(System.in);
    private static Scanner input1 = new Scanner(System.in);
    private static Scanner input2 = new Scanner(System.in);
    public static void main(String[] args) {
        //Creating Mongo Client
        MongoClient mongoClient = new MongoClient("localhost",27017);

        //Creating credentials
        MongoCredential credential =
MongoCredential.createCredential("Akshaan","VehicleDatabase","akshaan123456".toCharAr
ray());

        System.out.println("Connected to Database successfully");

        //Accessing Database
        MongoDatabase database = mongoClient.getDatabase("VehicleDatabase");
//creates the database

        System.out.println("Credentials ::"+ credential);

        //database.createCollection("Vehicles");

        System.out.println("Connection created successfully");

        MongoCollection<Document> collectionVehicles =
database.getCollection("Vehicles"); //creates the collection Vehicles
        MongoCollection<Document> collection1 = database.getCollection("Cars");
//creates the collection Car
        MongoCollection<Document> collection2 = database.getCollection("Motorbikes");
//creates the collection Motorbikes

        System.out.println("Collection selected successfully");
        LocalDate localDate = LocalDate.now();
        LocalTime localTime = LocalTime.now();
        System.out.println("~~~~~ Accessing on "+ localDate+" at "+localTime+"
~~~~~");

        System.out.println("Select option:");
        System.out.println("  1) Add Vehicle");
        System.out.println("  2) Delete Vehicle");
        System.out.println("  3) Print List of items");
        System.out.println("  4) Save");
        System.out.println("  5) Generate Report of the recent changes");
        System.out.println("  6) Search Vehicle");
        System.out.println("  7) Exit");
        System.out.print("Enter option: ");
        int option = getInteger(input);
        while (!(option>=1 && option<=7)){
            System.err.println("Invalid input!");
            System.out.print("Please reenter: ");
            option = getInteger(input);
        }
```

```
        while (true) {
            switch (option) {
                case 1:
                    System.out.println("---Add Vehicle---");
                    System.out.println("Select type of vehicle to be added: ");
                    System.out.println("  1) Car");
                    System.out.println("  2) Motorbike");
                    System.out.print("Enter type: ");
                    int type = getInteger(input);
                    while (!(type==1 || type==2)){
                        System.err.println("Invalid input!");
                        System.out.print("Please reenter: ");
                        type = getInteger(input);
                    }
                    if (type == 1) {
                        System.out.print("Plate number: ");
                        String plateNumber = input1.nextLine();
                        while (!(plateNumber.length()==8 || plateNumber.length()==7||
plateNumber.length()==6)){
                            System.err.println("Invalid plate number!");
                            System.out.print("Please renter: ");
                            plateNumber = input1.nextLine();
                        }
                        System.out.print("Make: ");
                        String make = input2.next();
                        System.out.print("Model: ");
                        String model = input1.next();
                        System.out.print("Color: ");
                        String color = input2.next();
                        System.out.print("Capacity (When entering type 'cc' at the
end): ");
                        String capacity = input2.next();
                        while (!capacity.contains("cc")){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            capacity = input1.next();
                        }
                        System.out.print("Transmission: ");
                        String transmission = input1.next();
                        while (!((transmission.equalsIgnoreCase("auto")||
(transmission.equalsIgnoreCase("manual"))))){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            transmission = input1.next();
                        }
                        System.out.print("Fuel Type: ");
                        String fuelType = input2.next();
                        while (!((fuelType.equalsIgnoreCase("diesel")||
(fuelType.equalsIgnoreCase("petrol"))))){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            fuelType = input1.next();
                        }
                        System.out.print("Year of Production: ");
                        int yearOfProduction = getInteger(input);
```

```
                        while (!(yearOfProduction<=2019)){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            yearOfProduction = input1.nextInt();
                        }
                        System.out.print("No. of doors: ");
                        int noOfDoors = getInteger(input1);
                        while (!(noOfDoors<6 && noOfDoors>1)){
                            System.err.println("Invalid input!. Cars cannot have "+
noOfDoors+ " of doors");
                            System.out.print("Please renter: ");
                            noOfDoors = input1.nextInt();
                        }
                        System.out.print("No. of Seats: ");
                        int noOfSeats = getInteger(input1);
                        while (!(noOfSeats<10 && noOfSeats>=2)){
                            System.err.println("Invalid input!. Cars cannot have "+
noOfSeats+ " of doors");
                            System.out.print("Please renter: ");
                            noOfSeats = input1.nextInt();
                        }
                        System.out.print("Hybrid status (If hybrid enter 'Hybrid',
else enter 'Non hybrid' : ");
                        String hybridStatus = input2.next();
                        while (!((hybridStatus.equalsIgnoreCase("hybrid")||
(hybridStatus.equalsIgnoreCase("nonhybrid"))))){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            hybridStatus = input1.next();
                        }
                        Document document1 = new Document();
                        document1.append("plateNumber", plateNumber);
                        document1.append("make", make);
                        document1.append("model", model);
                        document1.append("color", color);
                        document1.append("capacity", capacity);
                        document1.append("transmission", transmission);
                        document1.append("fuelType", fuelType);
                        document1.append("yearOfProduction",yearOfProduction);
                        document1.append("noOfDoors",noOfDoors);
                        document1.append("noOfSeats",noOfSeats);
                        document1.append("hybridStatus",hybridStatus);
                        collection1.insertOne(document1); //adds items to the mongodb
database
                        collectionVehicles.insertOne(document1);
                        Vehicle vehicle = new Car(plateNumber, make, model, color,
capacity, transmission, fuelType, yearOfProduction, noOfDoors,noOfSeats,
hybridStatus);
                        manager.addVehicle(vehicle);
                    } else {
                        System.out.print("Plate number: ");
                        String plateNumber = input1.nextLine();
                        while (!(plateNumber.length()==8 || plateNumber.length()==7||
plateNumber.length()==6)){
                            System.err.println("Invalid plateNumber!");
```

```
                            System.out.print("Please renter: ");
                            plateNumber = input1.nextLine();
                        }
                        System.out.print("Make: ");
                        String make = input1.next();
                        System.out.print("Model: ");
                        String model = input1.next();
                        System.out.print("Color: ");
                        String color = input2.next();
                        System.out.print("Capacity (When entering type 'cc' at the
end): ");
                        String capacity = input2.next();
                        while (!capacity.contains("cc")){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            capacity = input1.next();
                        }
                        System.out.print("Transmission: ");
                        String transmission = input1.next();
                        while (!((transmission.equalsIgnoreCase("auto")||
(transmission.equalsIgnoreCase("manual"))))){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            transmission = input1.next();
                        }
                        System.out.print("Fuel Type: ");
                        String fuelType = input2.next();
                        while (!((fuelType.equalsIgnoreCase("diesel")||
(fuelType.equalsIgnoreCase("petrol"))))){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            fuelType = input1.next();
                        }
                        System.out.print("Year of Production: ");
                        int yearOfProduction = getInteger(input);
                        while (!(yearOfProduction<=2019)){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            yearOfProduction = input1.nextInt();
                        }
                        System.out.print("Start Type (Enter 'Push' if push start,
else 'Key' if key start): ");
                        String startType = input2.next();
                        while (!((startType.equalsIgnoreCase("push")||
(startType.equalsIgnoreCase("key"))))){
                            System.err.println("Invalid input!");
                            System.out.print("Please renter: ");
                            startType = input1.next();
                        }
                        System.out.print("Bike Type (Enter 'Scooty' if scooty type
bike, else 'Normal' if not scooty type): ");
                        String bikeType = input1.next();
                        while (!((bikeType.equalsIgnoreCase("scooty")||
(bikeType.equalsIgnoreCase("normal"))))){
                            System.err.println("Invalid input!");
```

```
                            System.out.print("Please renter: ");
                            bikeType = input1.next();
                        }
                        Document document2 = new Document();
                        document2.append("plateNumber", plateNumber);
                        document2.append("make", make);
                        document2.append("model", model);
                        document2.append("color", color);
                        document2.append("capacity", capacity);
                        document2.append("transmission", transmission);
                        document2.append("fuelType", fuelType);
                        document2.append("yearOfProduction",yearOfProduction);
                        document2.append("startType",startType);
                        document2.append("bikeType",bikeType);
                        collection2.insertOne(document2); //adds items to the mongodb
database
                        collectionVehicles.insertOne(document2);
                        Vehicle vehicle = new Motorbike(plateNumber, make, model,
color, capacity, transmission, fuelType, yearOfProduction, startType, bikeType);
                        manager.addVehicle(vehicle);
                    }
                    break;
                case 2:
                    System.out.print("Enter plate number: ");
                    String plateNumber = input2.nextLine();
                    boolean found = false;
                    for (Vehicle vehicle : manager.getVehicles()) {
                        if (plateNumber.equals(vehicle.getPlateNumber())) {
                            manager.deleteVehicle(vehicle);
                            found = true;
                            break;
                        }
                    }
                    if(!found){
                        System.out.println("Vehicle with the plate number:
"+plateNumber+" not found!");
                    }

collectionVehicles.deleteOne(Filters.eq("plateNumber",plateNumber));
                    collection1.deleteOne(Filters.eq("plateNumber",plateNumber));
                    collection2.deleteOne(Filters.eq("plateNumber",plateNumber));
                    break;
                case 3:
                    if(manager.getVehicles().size()==0){
                        System.out.println("---No Vehicles!---");
                    }else {
                        System.out.println("-----Vehicles-----");
                        System.out.println("----------Recently added----------");
//prints recently added vehicles
                        manager.getVehicles().sort(new MakeComparator());
                        for (Vehicle vehicle : manager.getVehicles()) {
                            System.out.println(vehicle);
                        }
                    }
                    System.out.println("-----------------------------------------
```

```
");
                    System.out.println("--------------All the Vehicles--------------
"); //prints all the vehicles
                    Document query1 = new Document();
                    List<Document> list1 = (List<Document>)
collectionVehicles.find(query1).into(
                            new ArrayList<Document>());
                    for (Document doc:list1) {
                        System.out.println(doc);
                    }
                    break;
                case 4:
                    manager.save();
                    break;
                case 5:
                    manager.generateReport();
                    break;
                case 6:
                    try {
                        URI uri = new URI("http://localhost:4400/search"); //inside
paranthesis is the url which it needs to be directed
                        java.awt.Desktop.getDesktop().browse(uri);
                    } catch (URISyntaxException | IOException e) {
                        System.out.println("Exception");
                    }
                    break;
                case 7:
                    System.out.println("---Program quits!---");
                    System.exit(0);
                default:
                    System.out.println("Select option:");
                    System.out.println("  1) Add Vehicle");
                    System.out.println("  2) Delete Vehicle");
                    System.out.println("  3) Print List of items");
                    System.out.println("  4) Save");
                    System.out.println("  5) Generate Report");
                    System.out.println("  6) Search Vehicle");
                    System.out.println("  7) Exit");
                    System.out.print("Enter option: ");
                    option = getInteger(input);
                    continue;
            }
            System.out.println("Select option:");
            System.out.println("  1) Add Vehicle");
            System.out.println("  2) Delete Vehicle");
            System.out.println("  3) Print List of items");
            System.out.println("  4) Save");
            System.out.println("  5) Generate Report of the recent changes");
            System.out.println("  6) Search Vehicle or View Vehicle/s");
            System.out.println("  7) Exit");
            System.out.print("Enter option: ");
            option = getInteger(input);
        }
    }
    private static int getInteger(Scanner input){ //this method validates whether the
```

```
input is an int or not
        while (!input.hasNextInt()){
            System.err.println("Invalid input!");
            System.out.print("Please reenter: ");
            input.next();
        }
        return input.nextInt();
    }
}
```

Angular + Spring boot

Angular Part

Default port of 4200 was changed to 4400

Index.html – It is main html page of this website.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>WestminsterRentalVehicle</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="/assets/favicon.png">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Style.css – It is the main style file of this website.

```
/* You can add global styles to this file, and also import other style files */
@import "../node_modules/@angular/material/prebuilt-themes/deeppurple-amber.css";
body{
  padding: 15px;
  background: ghostwhite;
}
```

No changes done for other files at main level

App Component

It is the root component of this project.

App.component.ts – It is the app component

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
```

```
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'rentAVehicle';
}
```

App.component.html – It is the html file associated with the app.component.ts

```
<a routerLink=""><img src="/assets/logo.jpg" width="350px" height="50px"></a>
<br><br><br><br>
<router-outlet></router-outlet>
```

App.component.css – It is the style file associated with the app.component.ts

```
h1 a{
  color: black;
  text-decoration: none;
}
```

App.module.ts – It is the file that has all the routing in the application.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { SearchResultsComponent } from './search-results/search-results.component';
import { AllVehiclesComponent } from './all-vehicles/all-vehicles.component';
import {Router, RouterModule,Routes} from "@angular/router";
import { NotFoundComponent } from './not-found/not-found.component';
import {FormsModule} from "@angular/forms";
import { LoginComponent } from './login/login.component';
import { LoginSignUpHomeComponent } from './login-sign-up-home/login-sign-up-
home.component';
import { SearchComponent } from './login-sign-up-home/search/search.component';
import { SignUpComponent } from './sign-up/sign-up.component';
import { ViewAllVehiclesComponent } from './login-sign-up-home/view-all-
vehicles/view-all-vehicles.component';
import {HttpClientModule} from "@angular/common/http";

import { MainServiceService } from './mainService/main-service.service';
import { VehicleServiceService } from './servicesForModels/Vehicle/vehicle-
service.service';
import { CarServiceService } from './servicesForModels/Car/car-service.service';
import { MotorbikeServiceService } from './servicesForModels/Motorbike/motorbike-
service.service';
import { ThankyouComponent } from './thankyou/thankyou.component';
import { MainBookingComponent } from './main-booking/main-booking.component';
import { ShowVehiclesComponent } from './show-vehicles/show-vehicles.component';
import {MatTableModule} from "@angular/material/table";
import { CarComponentComponent } from './show-vehicles/car-component/car-
component.component';
```

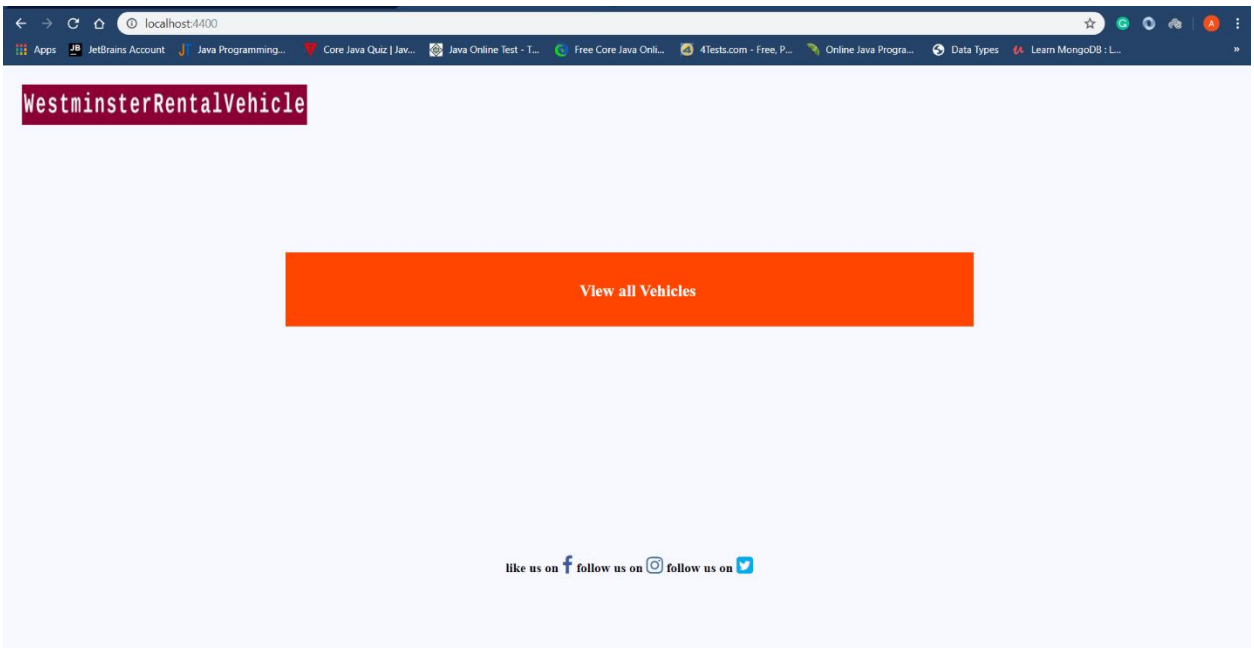
```
import { MotorbikeComponentComponent } from './show-vehicles/motorbike-
component/motorbike-component.component';
import {VehicleSelectionComponent} from "./vehicle-selection/vehicle-
selection.component";
import {MatDialogModule,MatFormFieldModule,MatInputModule} from "@angular/material";
import {BrowserAnimationsModule} from '@angular/platform-browser/animations';
import { from } from 'rxjs';
import {CarService} from "src/app/mainService/Car/car.service";
import { MotorbikeService } from './mainService/Motorbike/motorbike.service';
import { MakeFilterCarPipe } from './make-filter-car.pipe';
import { MakeFilterMotorbikePipe } from './make-filter-motorbike.pipe';
import { CapcityFilterPipe } from './vehicle-selection/capcity-filter.pipe';
import { PlateNumberFilterPipe } from './search-results/plateNumberFilter/plate-
number-filter.pipe';
import { PlateNumberCarFilterPipe } from './search-
results/plateNumberFilter/carplateNumberFilter/plate-number-car-filter.pipe';
import { PlateNumberMotorbikeFilterPipe } from './search-
results/plateNumberFilter/motorbikeplateNumberFilter/plate-number-motorbike-
filter.pipe';
const appRoutes:Routes = [
  {
    path:"",
    component: HomeComponent
  },
  {
    path:"search",
    component: SearchResultsComponent
  },
  {
    path:"viewAllVehicles",
    component: AllVehiclesComponent
  },
  {
    path:"login",
    component: LoginComponent
  },
  {
    path:"signUp",
    component: SignUpComponent
  },
  {
    path:"loginSignUp-home",
    component:LoginSignUpHomeComponent
  },
  {
    path:"loginSignUp-home/search",
    component:SearchComponent
  },
  {
    path:"loginSignUp-home/viewVehicles",
    component:ViewAllVehiclesComponent
  },
  {
    path:"thankyou",
    component:ThankyouComponent
```

```
  },
  {
    path:"booking",
    component:MainBookingComponent
  },
  {
    path:"car",
    component:CarComponentComponent,
    data:{title:"Cars"}
  },
  {
    path:"motorbike",
    component:MotorbikeComponentComponent
  },
  {
    path:"booking/:plateNumber",
    component:MainBookingComponent
  },
  {
    path:"**",
    component: NotFoundComponent
  }
];
@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    SearchResultsComponent,
    AllVehiclesComponent,
    NotFoundComponent,
    LoginComponent,
    LoginSignUpHomeComponent,
    SearchComponent,
    SignUpComponent,
    ViewAllVehiclesComponent,
    ThankyouComponent,
    MainBookingComponent,
    ShowVehiclesComponent,
    CarComponentComponent,
    MotorbikeComponentComponent,
    VehicleSelectionComponent,
    MakeFilterCarPipe,
    MakeFilterMotorbikePipe,
    MainBookingComponent,
    CapcityFilterPipe,
    PlateNumberFilterPipe,
    PlateNumberCarFilterPipe,
    PlateNumberMotorbikeFilterPipe
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    RouterModule.forRoot(appRoutes,{
      enableTracing:true
    }),
```

```
    FormsModule,
    HttpClientModule,
    MatTableModule,
    MatDialogModule,
    BrowserAnimationsModule,
    MatFormFieldModule,
    MatInputModule,
  ],
  providers: [MainServiceService,VehicleServiceService,CarService,MotorbikeService],
  bootstrap: [AppComponent],

entryComponents:[VehicleSelectionComponent,CarComponentComponent,MotorbikeComponentCo
mponent]
})
export class AppModule { }
```

Home component

It is the home page of the application.



Home.component.ts – It is the home component

```
import { Component, OnInit } from '@angular/core';
import { MainServiceService } from '../mainService/main-service.service';
import { Vehicle } from '../models/Vehicle';
import {MatDialog,MatDialogConfig} from "@angular/material";
import {VehicleSelectionComponent} from "../vehicle-selection/vehicle-
selection.component";

import { from } from 'rxjs';
@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
```

```
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  allVehicles:Vehicle;

  constructor(private service:MainServiceService,private dialog:MatDialog) { }

  ngOnInit() {
  }

  printAllVehicles(){
    console.log(this.service.allVehicles());
    this.service.allVehicles().subscribe(
      response=> this.handleResponse(response)
    );

  }
  handleResponse(response){
    console.log(response)
    this.allVehicles = response.message;
  }

  selection(){
    const dialogConfig = new MatDialogConfig();
    dialogConfig.disableClose = false;
    dialogConfig.autoFocus = true;
    dialogConfig.width="25%",
    this.dialog.open(VehicleSelectionComponent,dialogConfig);
  }

}
```

Home.component.html – It is the html file associated with the home component.

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<body>
<!-- <div class="buttons">
<button routerLink="/login">Login</button>
<button routerLink="/signUp">Sign Up</button>
</div> -->
<br><br><br><br>
<div class="content">
  <!-- <div id="vertical"></div> -->
  <ul>
    <!-- <li><a routerLink="/search">Search Vehicle</a></li> -->
    <li><a style="cursor: pointer;"><div id="div1" (click)="selection()" >View all
Vehicles</div></a></li>
  </ul>
</div>
<div class="div">
{{allVehicles}}
</div>
<div style="margin-top: 280px;text-align: center;">
```

```
<footer>
    <p><b>like us on <a href="#" style="color:#3b5998;"><i class="fa fa-
facebook"style="font-size:24px"></i></a>
       follow us on <a href="#" style="color:#3f729b;"><i class="fa fa-
instagram"style="font-size:24px"></i></a>
        follow us on <a href="#" style="color:#00acee;"><i class="fa fa-twitter-
square"style="font-size:24px"></i></a></b></p>

</footer>
</div>
</body>
```

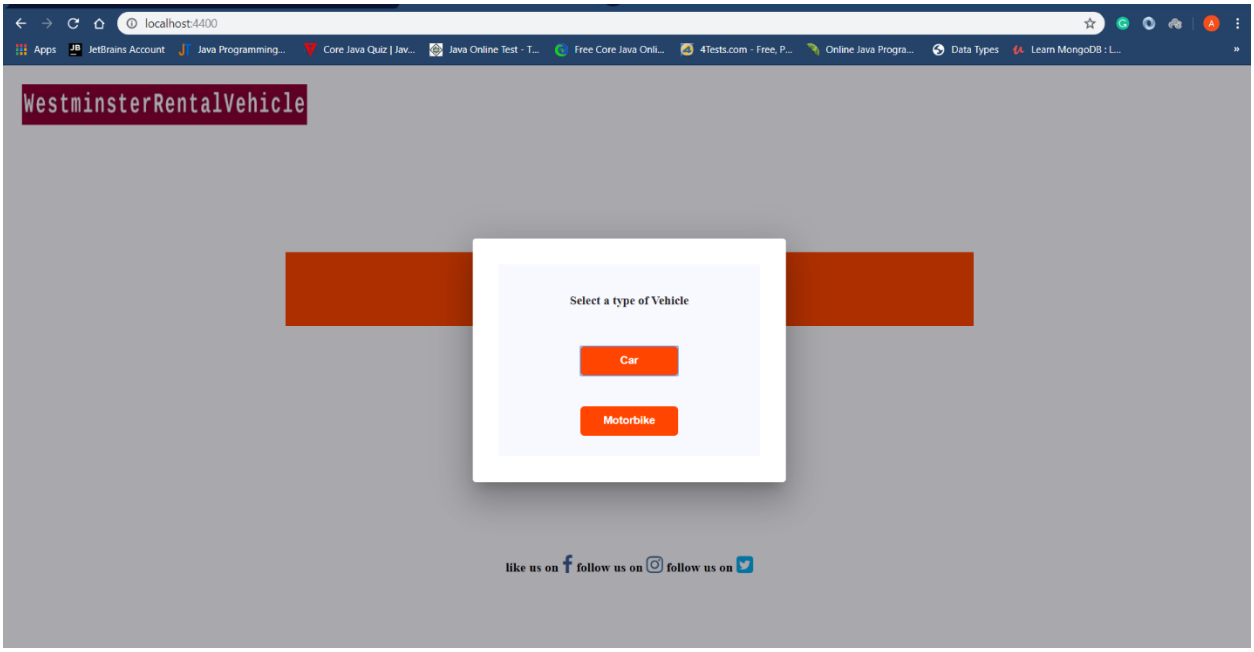
Home.component.css – It is the style file associated with the home component.

```
button{
  background: orangered;
  color: white;
  font-weight: bold;
  border: orangered;
  padding: 10px;
  border-radius: 5px;
  margin: 10px;
  cursor: pointer;
  width: 120px;
}
.buttons{
  text-align: right;
}
.content{
  background: orangered;
  padding: 20px;
  margin-left:300px ;
  margin-right: 300px;
  padding-left:210px ;
  color: white;
  font-weight: bold;
  height: 50px;
}
.content ul li{
  list-style-type: none;
  float: left;
  width: 365px;
  text-align: center;
}
.content ul li a{
  display: block;
  color: white;
  font-weight: bold;
  text-decoration: none;
  font-size: 20px;
}
#vertical{
  border-left: 1px solid white;
  height: 50px;
```

```
  position:absolute;
  left: 50%;
}

#div1{
  position: relative;
}
.div{
color: black;
}
```

Vehicle-Selection component

It is the page that the user can select a type of vehicle to view or book.



Vehicle-Selection.ts - It is the Vehicle-Selection component

```
import { Component, OnInit } from '@angular/core';
import {MatDialogRef} from "@angular/material";

@Component({
  selector: 'app-vehicle-selection',
  templateUrl: './vehicle-selection.component.html',
  styleUrls: ['./vehicle-selection.component.css']
})
export class VehicleSelectionComponent implements OnInit {

  constructor(public dialogBox:MatDialogRef<VehicleSelectionComponent>) { }

  ngOnInit() {
  }
```

```
  closeOne(){
    this.dialogBox.close();
  }
  closeTwo(){
    this.dialogBox.close();
  }

}
```

Vehicle-Selection.html – It is the html file associated with the Vehicle-Selection component.

```
<body>
<div class="d-flex justify-content-between">
    <h4>Select a type of Vehicle</h4>
    <br>
    <button routerLink="/car" (click)="closeOne()">Car</button>
    <br><br>
    <button routerLink="/motorbike" (click)="closeTwo()">Motorbike</button>

    <!-- <button mat-button (click)="close()">
        <mat-icon>close</mat-icon>
    </button> -->
</div>
</body>
```

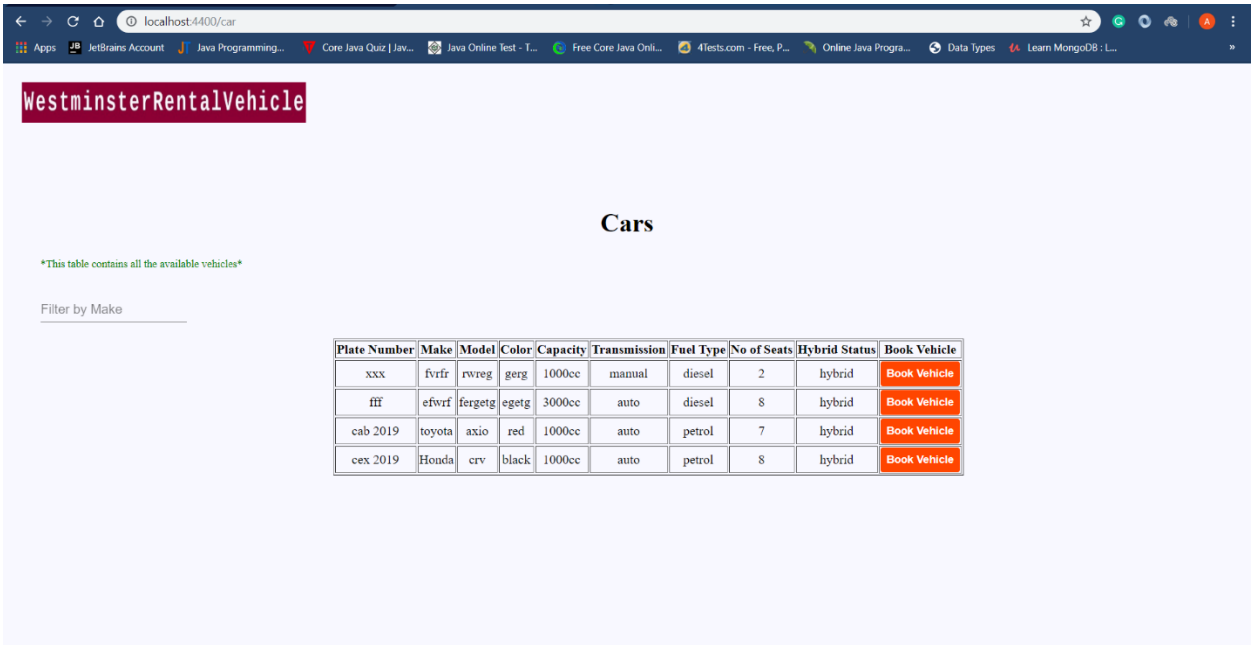
Vehicle-Selection.css – It is the style file associated with the Vehicle-Selection component.

```
body{
    text-align: center;
    font-weight: bold;
    font-size: 15px;
}
button{
    background: orangered;
  color: white;
  font-weight: bold;
  border: orangered;
  padding: 10px;
  border-radius: 5px;
  margin: 10px;
  cursor: pointer;
  width: 120px;
}
```

When clicked Car button it'll direct to the Car Component, when clicked Motorbike button it'll direct to the Motorbike Component

Car component

It is the page that the user can view all the available cars.



Car-Component.Component.ts - It is the Car-Component.

```
import { Component, OnInit } from '@angular/core';
import { Observable, from } from 'rxjs';
import {CarService} from "src/app/mainService/Car/car.service";
import {Car} from "src/app/models/Car";
import {MatTableDataSource} from '@angular/material/table';
import {Router} from "@angular/router";
import {ActivatedRoute} from "@angular/router";



@Component({
  selector: 'app-car-component',
  templateUrl: './car-component.component.html',
  styleUrls: ['./car-component.component.css'],
  providers:[CarService]
})
export class CarComponentComponent implements OnInit {
  cars:Car[];
  search:String;


  constructor(private carService:CarService,private router:Router,private
route:ActivatedRoute) { }

  ngOnInit() {
    this.carService.getCars().subscribe(
      response => {
        console.log(response);
        this.cars = response;
      }
```

```
    );


  }
  handleSuccessfulResponse(response)
{
    this.cars=response;
}


dataSource = new MatTableDataSource(this.cars);

}
```

Car-Component.Component.html – It is the html file associated with the Car-Component.

```
<body>
<h1 style="text-align: center;">Cars</h1>
<small style="color: green;">*This table contains all the available
vehicles*</small><br><br>
<mat-form-field>
  <input matInput  placeholder="Filter by Make" [(ngModel)]="search">
</mat-form-field><br>
<div style="text-align: center;margin-left: 360px;">
<table border="1">
    <thead></thead>
    <tr>
      <th>Plate Number</th>
      <th>Make</th>
      <th>Model</th>
      <th>Color</th>
      <th>Capacity</th>
      <th>Transmission</th>
      <th>Fuel Type</th>
      <th>No of Seats</th>
      <th>Hybrid Status</th>
      <th>Book Vehicle</th>
    </tr>

    <tbody>
        <tr *ngFor="let car of cars | makeFilterCar:search">
            <td>{{car.plateNumber}}</td>
            <td>{{car.make}}</td>
            <td>{{car.model}}</td>
            <td>{{car.color}}</td>
            <td>{{car.capacity}}</td>
            <td>{{car.transmission}}</td>
            <td>{{car.fuelType}}</td>
            <td>{{car.noOfSeats}}</td>
            <td>{{car.hybridStatus}}</td>
            <td><button id="book" [routerLink]="['/booking', car.plateNumber]">Book
Vehicle</button></td>
        </tr>
    </tbody>
  </table>
```

```
</div>
</body>
```

Car-Component.Component.css – It is the style file associated with the Car-Component.

```
#book{
    background-color: orangered;
    color: white;
    font-weight:bold ;
    border:2px solid orangered;
    padding: 5px;
    border-radius:3px ;
    cursor: pointer;
}
```

Car-Service.ts – It is the typescript file which has the connection with the backend api.

```
import { Injectable } from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {Car} from "src/app/models/Car";
import {Observable} from "rxjs";
import { CarBooking } from 'src/app/models/CarBooking';

@Injectable({
  providedIn: 'root'
})
export class CarService {
  carData:Car;
  plateNumber:string

  constructor(private http:HttpClient) { }

  getCars(){
    return this.http.get<Car[]>("http://localhost:8080/cars");
  }

  getCar():Observable<Car>{
    return this.http.get<Car>("http://localhost:8080/cars/"+this.plateNumber);
  }


addCarBooking(carBooking:CarBooking,email:string,nic:string,pickupdate:Date,dropOffDa
te:Date){
     return
this.http.post("http://localhost:8080/bookingcar/"+this.getCar()+"/"+email+"/"+nic+"/
"+pickupdate+"/"+dropOffDate,carBooking);
  }
}
```

Car.ts – It is the typescript file which has the sample model like the Car class in java project.

```
import { Vehicle } from './Vehicle';
import {Observable} from "rxjs";
```

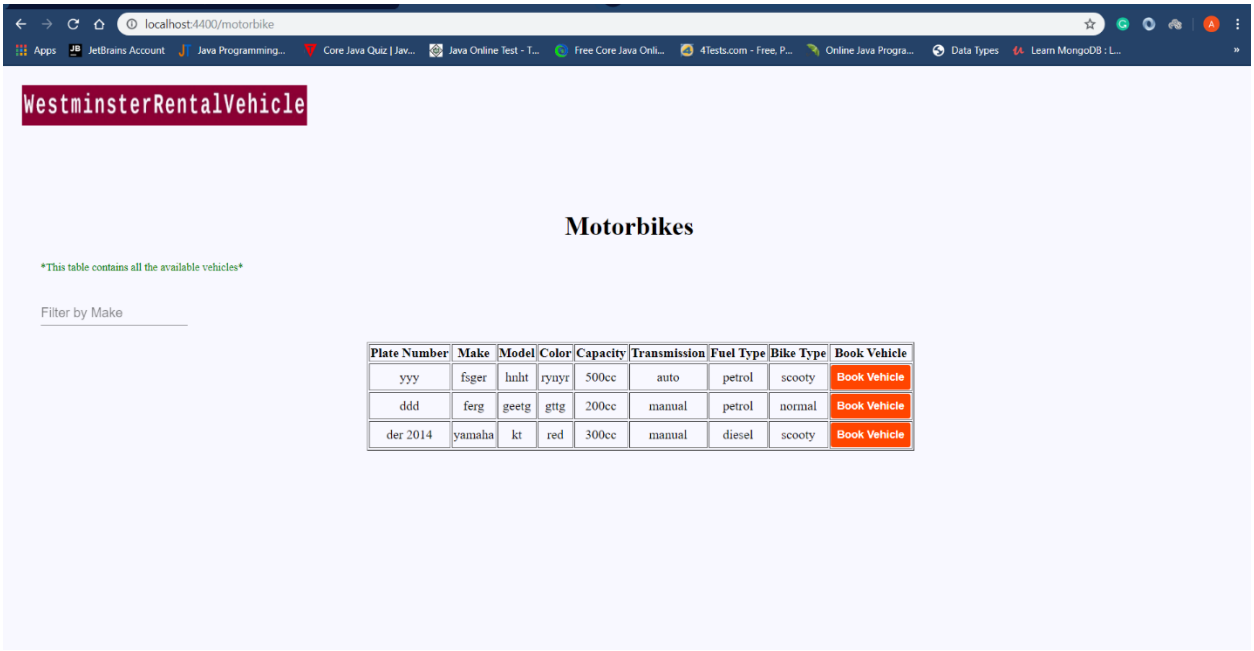
```
export class Car{
    constructor(public plateNumber:String,public make:String,public
model:String,public color:String,public capacity:String,public
transmission:String,public fuelType:String,public yearOfProduction,public
noOfDoors:number,public noOfSeats:number,public hybridStatus:string){

    }
}
```

Make-Filter-Car.pipe.ts – For filtering of the cars in Car component by make of the cars.

```
import { Pipe, PipeTransform } from '@angular/core';
import {Car} from "src/app/models/Car";


@Pipe({
  name: 'makeFilterCar'
})
export class MakeFilterCarPipe implements PipeTransform {

  transform(cars:Car[],search:string):Car[] {
    if(!cars || !search){
      return cars;
    }
    return cars.filter(car => car.make.toLowerCase().indexOf(search.toLowerCase())
!==-1);
  }

}
```

Motorbike component

It is the page that the user can view all the available motorbikes.



Motorbike-Component.Component.ts - It is the Motorbike-Component.

```
import { Component, OnInit } from '@angular/core';
import { Observable, from } from 'rxjs';
import {Motorbike} from "src/app/models/Motorbike";
import { MotorbikeService } from 'src/app/mainService/Motorbike/motorbike.service';

@Component({
  selector: 'app-motorbike-component',
  templateUrl: './motorbike-component.component.html',
  styleUrls: ['./motorbike-component.component.css']
})
export class MotorbikeComponentComponent implements OnInit {
  motorbikePlateNumber:string;
  motorbikes:Motorbike[];
  search:string;
  plateNumber:string;


  constructor(private motorbikeServices?:MotorbikeService,) { }

  ngOnInit() {
    this.motorbikeServices.getMotorbike().subscribe(
      response => {
        console.log(response);
        this.motorbikes = response;
      }
    );

  }
  handleSuccessfulResponse(response)
{
    this.motorbikes=response;
}
```

```
  public get motorbikePlateNum():any{
    return this.motorbikePlateNumber;
  }

}
```

Motorbike-Component.Component.html – It is the html file associated with the Motorbike Component.

```
<body>
<h1 style="text-align: center;">Motorbikes</h1>
<small style="color: green;">*This table contains all the available
vehicles*</small><br><br>
<mat-form-field>
  <input matInput  placeholder="Filter by Make" [(ngModel)]="search">
</mat-form-field><br>
<div style="text-align: center;margin-left: 400px;">
    <table border="1" style="width: 100%;" style="text-align: center;">
        <thead></thead>
        <tr>
          <th>Plate Number</th>
          <th>Make</th>
          <th>Model</th>
          <th>Color</th>
          <th>Capacity</th>
          <th>Transmission</th>
          <th>Fuel Type</th>
          <th>Bike Type</th>
          <th>Book Vehicle</th>
        </tr>

        <tbody>
            <tr *ngFor="let motorbike of motorbikes | makeFilterMotorbike:search">
                <td [(ngModel)]="motorbikePlateNumber">{{motorbike.plateNumber}}</td>
                <td>{{motorbike.make}}</td>
                <td>{{motorbike.model}}</td>
                <td>{{motorbike.color}}</td>
                <td>{{motorbike.capacity}}</td>
                <td>{{motorbike.transmission}}</td>
                <td>{{motorbike.fuelType}}</td>
                <td>{{motorbike.bikeType}}</td>
                <td><button id="book" [routerLink]="['/booking',
motorbike.plateNumber]">Book Vehicle</button></td>
            </tr>
        </tbody>
      </table>
    </div>
    </body>
```

Motorbike-Component.Component.css – It is the style file associated with the Motorbike-Component.

```
#book{
    background-color: orangered;
    color: white;
    font-weight:bold ;
    border:2px solid orangered;
    padding: 5px;
    border-radius:3px ;
    cursor: pointer;
}
```

Motorbike-Service.ts – It is the typescript file which has the connection with the backend api.

```
import { Injectable } from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {Observable} from "rxjs";
import {Motorbike} from "../../models/Motorbike";

@Injectable({
  providedIn: 'root'
})
export class MotorbikeService {
  motorbikeData:Motorbike;

  constructor(private http:HttpClient) { }
  getMotorbike(){
    return this.http.get<Motorbike[]>("http://localhost:8080/motorbikes");
  }
}
```

Motorbike.ts – It is the typescript file which has the sample model like the Motorbike class in java file.

```
import { Vehicle } from './Vehicle';

export class Motorbike{
    plateNumber:String;
    make:String;
    model:String;
    color:String;
    capacity:String;
    transmission:String;
    fuelType:String;
    startType:string;
    bikeType:string;
}
```

Make-Filter-Motorbike.pipe.ts – For filtering of the motorbikes in Motorbike component by make of the motorbikes.

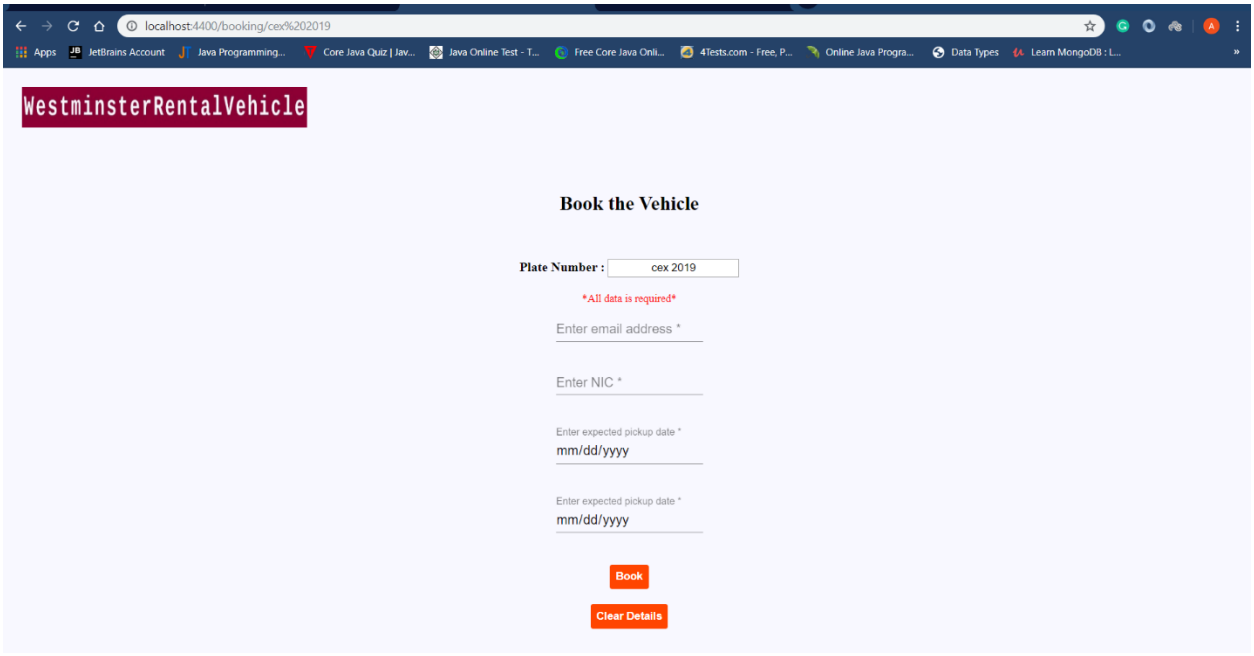
```
import { Pipe, PipeTransform } from '@angular/core';
import {Motorbike} from "src/app/models/Motorbike";
```

```
@Pipe({
  name: 'makeFilterMotorbike'
})
export class MakeFilterMotorbikePipe implements PipeTransform {

  transform(cars:Motorbike[],search:string):Motorbike[] {
    if(!cars || !search){
      return cars;
    }
    return cars.filter(car => car.make.toLowerCase().indexOf(search.toLowerCase())
!==-1);
  }

}
```

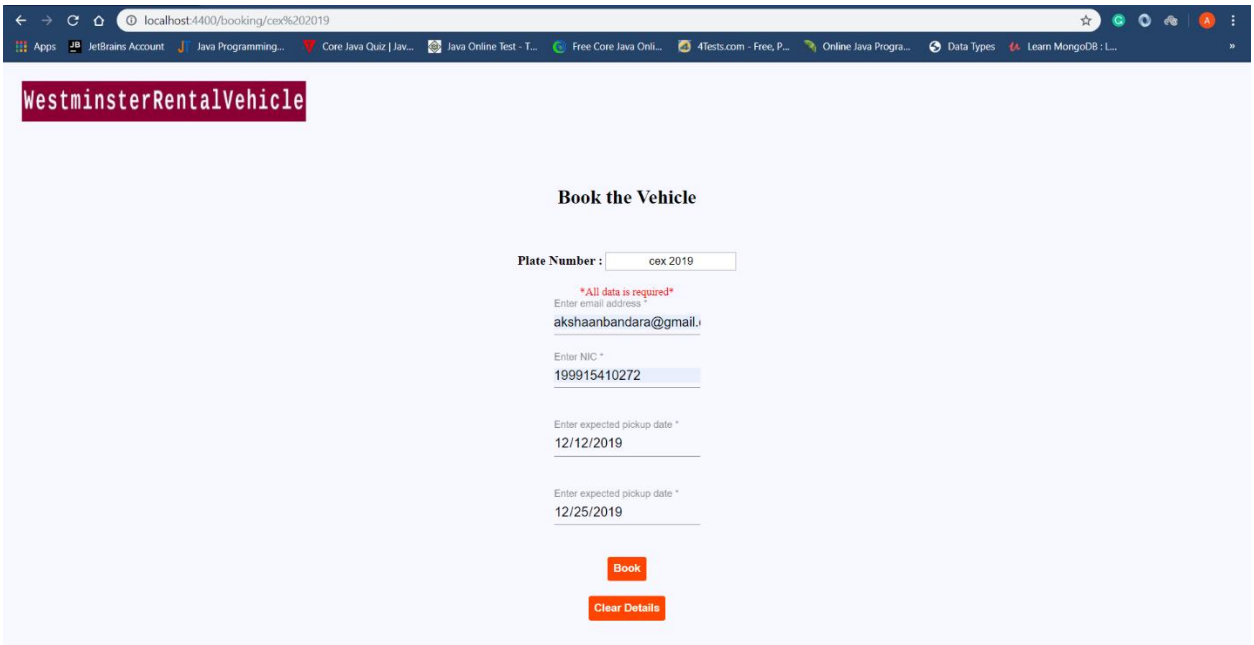
Main Booking component

The user selects a vehicle to be booked by clicking the book button in Car and Motorbike Components respectively. Then the user will be directed to the main-booking component. After filling the requirements, the user can book a vehicle.

Before entering details;



After entering details;



Router Parameters used to get the plate number form Car or Motorbike Components to Booking Component

Main-booking.Component.ts - It is the Booking Component

```
import { Component, OnInit } from '@angular/core';
import {Car} from "src/app/models/Car";
import {MatDialog,MatDialogConfig, MatDialogRef} from "@angular/material";
import {NgForm} from "@angular/forms";
import {MotorbikeComponentComponent} from "../show-vehicles/motorbike-
component/motorbike-component.component";
import { HttpClient } from 'selenium-webdriver/http';
import { MotorbikeService } from '../mainService/Motorbike/motorbike.service';
import {ActivatedRoute} from "@angular/router";
import { CarService } from '../mainService/Car/car.service';
import { CarBooking } from '../models/CarBooking';
import { Booking } from '../models/Booking';
import {BookingServiceService} from "../mainService/booking-service.service";
import { from } from 'rxjs';
import { MotorbikeBooking } from '../models/MotorbikeBooking';

@Component({
  selector: 'app-main-booking',
  templateUrl: './main-booking.component.html',
  styleUrls: ['./main-booking.component.css']
})

export class MainBookingComponent implements OnInit {
  car:Car;
  public plateNumber:string;
  email:string;
  nic:string;
```

```
  pickUpDate:Date;
  dropOffDate:Date;



  constructor(private route:ActivatedRoute,private bookingService:
BookingServiceService) {

   }

   booking:Booking = new
Booking(this.plateNumber,this.email,this.nic,this.pickUpDate,this.dropOffDate);

   carBooking:CarBooking = new
CarBooking(this.booking.plateNumber,this.booking.email,this.booking.nic,this.booking.
pickUpdate,this.booking.dropOffDate);

  //  motorbikeBooking:MotorbikeBooking = new
MotorbikeBooking(this.plateNumber,this.booking.email,this.booking.nic,this.booking.pi
ckUpdate,this.booking.dropOffDate);

  ngOnInit() {
    let platenumber = this.route.snapshot.paramMap.get('plateNumber');
    this.plateNumber = platenumber;
  }


  // onSubmit(){
  //   this.BookingServiceService.bookCar(this.booking).subscribe(
  //     data=>console.log("Success",data),
  //     error=>console.log("Error!",error)
  //   )
  // }

  onSubmit(){
    this.bookingService.bookCar(this.carBooking).subscribe(
      data=>{
        alert("Booking Successfull!")
        window.location.href="http://localhost:4400/thankyou"
      }
    );
    // this.bookingService.bookMotorbike(this.motorbikeBooking).subscribe(
    //   data=>{
    //     alert("Booking Successfull!")
    //   }
    // );
  }

}
```

Main-booking.Component.html – It is the html associated with the Booking Component

```
<body style="padding: 0px;margin: 0px;">
<h2 style="text-align: center;">Book the Vehicle</h2>
```

```
<br>
<form #bookingForm="ngForm" (ngSubmit)="onSubmit()">
<b><p style="text-align: center;">Plate Number : <input style="text-align: center;"
type="text" name="plateNumber" value="{{plateNumber}}" readonly/></p></b>
<div style="text-align: center;">
<small style="color: red;">*All data is required*</small><br>
<mat-form-field>
    <input matInput type="email" placeholder="Enter email address"
[(ngModel)]="booking.email" name="email" required/>
</mat-form-field><br>
<mat-form-field>
    <input matInput type="text" #nic = "ngModel" pattern="^\d{12}$" [class.is-
invalid] = "nic.invalid && nic.touched" class="form-control" placeholder="Enter NIC"
[(ngModel)]="booking.nic" name="nic" required/>
</mat-form-field><br><br>
<mat-form-field>
    <input matInput type="date" placeholder="Enter expected pickup date"
[(ngModel)]="booking.pickUpdate" name="pickupDate" required/>
</mat-form-field><br><br>
<mat-form-field>
    <input matInput type="date" placeholder="Enter expected pickup date"
[(ngModel)]="booking.dropOffDate" name="dropoffdate" required/>
</mat-form-field><br><br>
    <button class="submit"
[disabled]="bookingForm.form.invalid">Book</button><br><br><button
class="submit"type="reset">Clear Details</button>
</div>
</form>
</body>
```

Main-booking.Component.css – It is the css associated with the Booking Component

```
.submit{
    background-color: orangered;
    color: white;
    font-weight:bold ;
    border:2px solid orangered;
    padding: 5px;
    border-radius:3px ;
    cursor: pointer;
}
```

Booking-Service-Service.ts – It is the typescript file which has the connection with the backend api.

```
import { Injectable } from '@angular/core';
import {HttpClient} from "@angular/common/http";
import { Motorbike } from '../models/Motorbike';
import { Car } from '../models/Car';
import {Observable, from} from "rxjs";
import { MotorbikeServiceService } from '../servicesForModels/Motorbike/motorbike-
service.service';
import { CarBooking } from '../models/CarBooking';
import {MotorbikeBooking} from "../models/MotorbikeBooking";
```

```
@Injectable({
  providedIn: 'root'
})
export class BookingServiceService {


  // url = 'http://localhost:8080/value';
  constructor(private http:HttpClient) {

  }

  // bookCar(car: Car){
  //   return this.http.post<Car>(this.url,car);
  // }

  // booMotorbike(motorbike:Motorbike){
  //   this.http.post<Motorbike>(this.url,motorbike);
  // }

  bookCar(carBooking:CarBooking){
    return this.http.post<CarBooking>("http://localhost:8080/bookingcar",carBooking);
  }

  bookMotorbike(motorBooking:MotorbikeBooking){
    return
this.http.post<MotorbikeBooking>("http://localhost:8080/bookingcar",motorBooking);
  }

}
```

Booking.ts – It is the model of the booking.

```
export class Booking{

    constructor(public plateNumber:string,public email:string,public
nic:string,public pickUpdate:Date,public dropOffDate:Date){

    }
}
```

CarBooking.ts – It is the model of the car booking.

```
export class CarBooking{
    constructor(public plateNumber:string,public email:string,public
nic:string,public pickUpDate:Date,public dropOffDate:Date){

    }
}
```

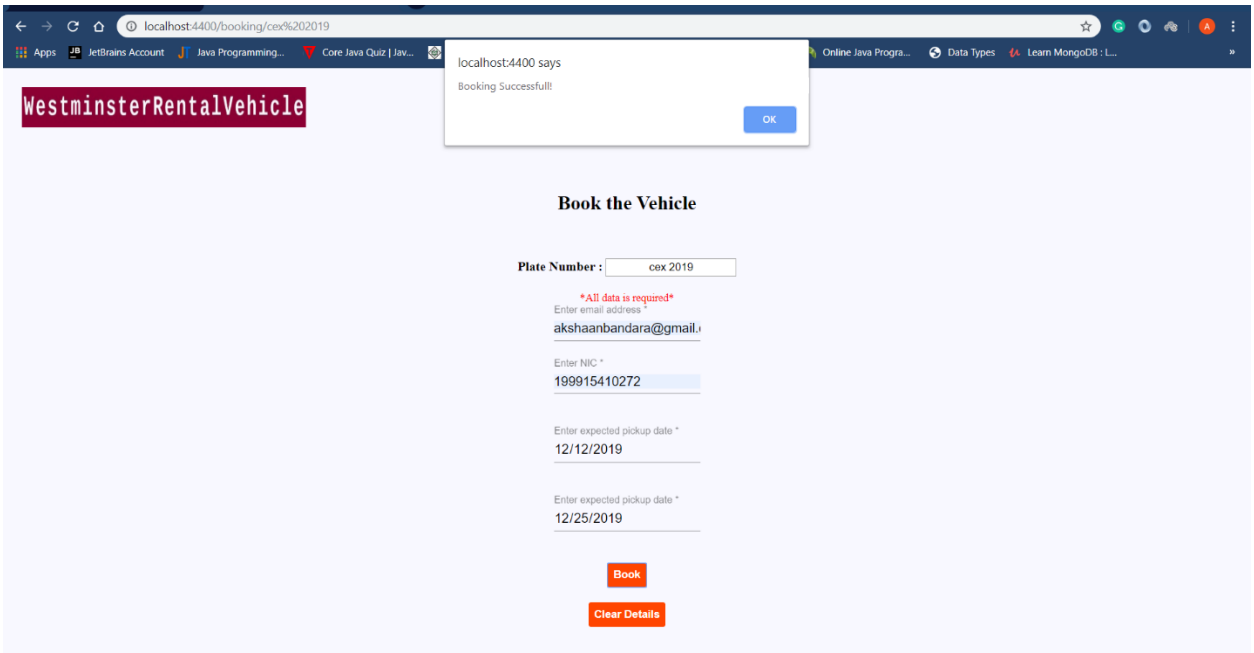
MotorbikeBooking.ts – It is the model of the motorbike booking.

```
export class MotorbikeBooking{
    constructor(public plateNumber:string,public email:string,public
nic:string,public pickupdate:Date,public dropOffDate:Date){

    }
}
```

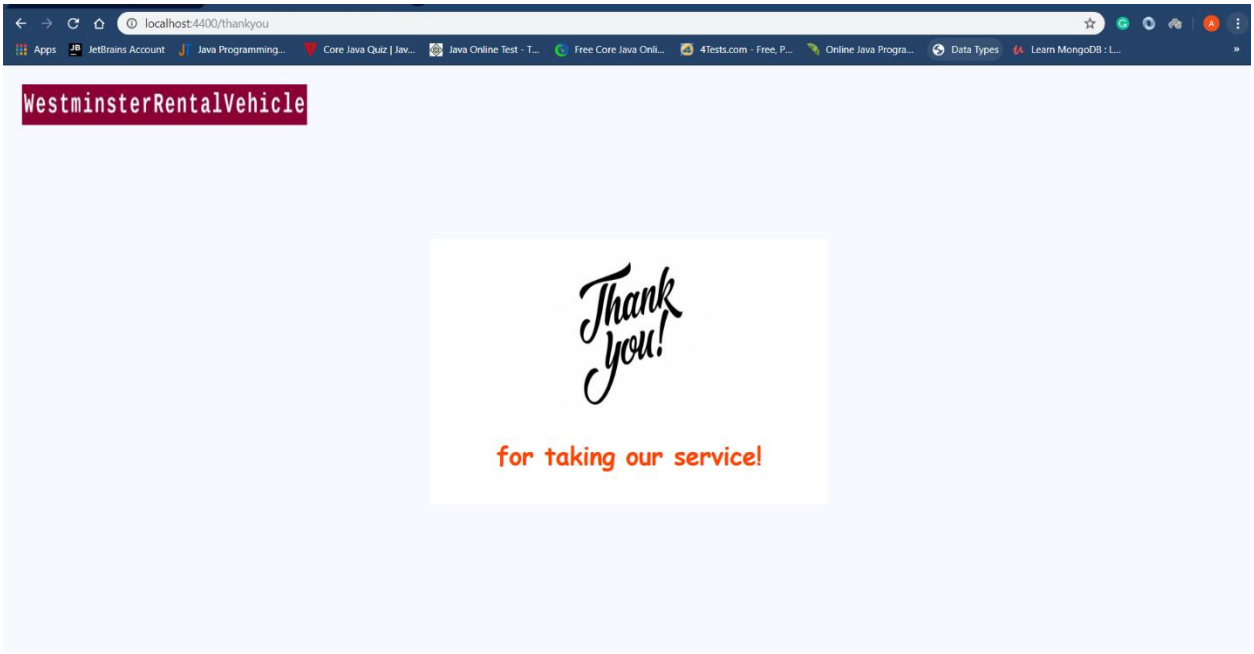
Book button will book the vehicle.

Clear Details button will clear the details that have been entered.

When the users enter Book button it displays that the booking is successful.



When the user clicks “OK” he/she will be directed to thankyou component.



Thankyou-Component,ts – It is the thank you component.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-thankyou',
  templateUrl: './thankyou.component.html',
  styleUrls: ['./thankyou.component.css']
})
export class ThankyouComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Thankyou-Component,html – It is the html associated with the Thank you Component.

```
<div class="div1">
    <img src="assets/thankyou.jpg" width="200" height="200"/><br>
    <p id="service">for taking our service!</p>
</div>
```

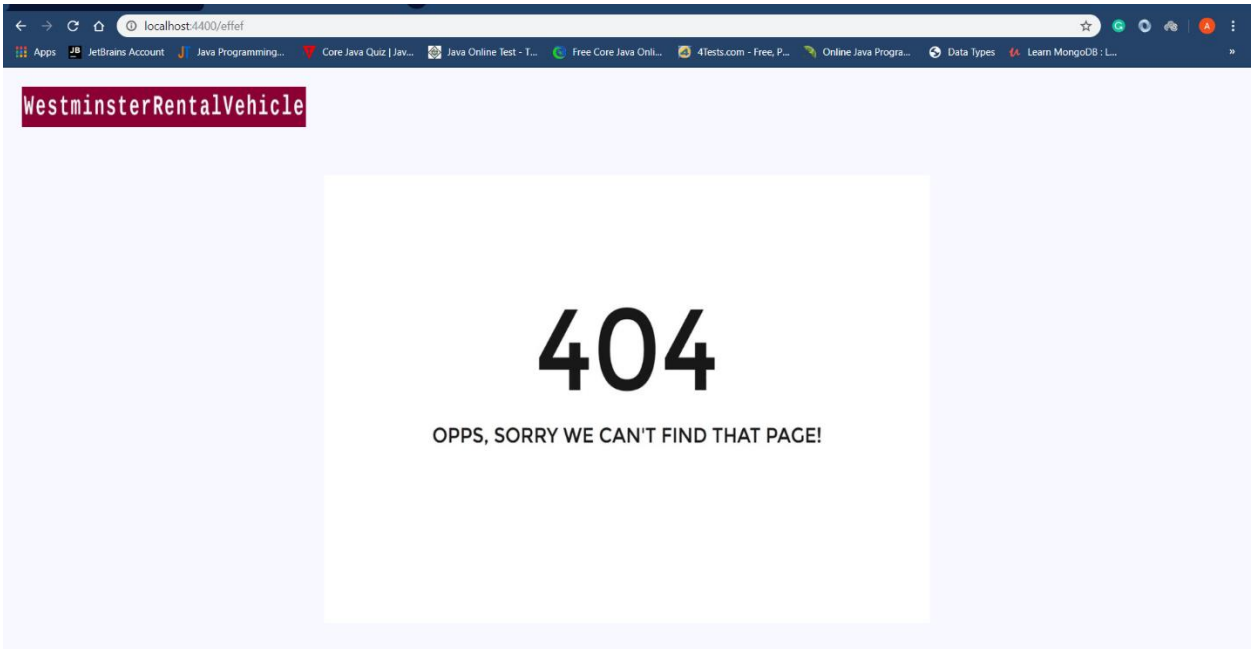
Thankyou-Component,css – It is the css associated with the Thank you Component.

```
.div1{
    text-align: center;
    padding: 10px;
```

```
    margin-top: 80px;
    margin-left: 500px;
    margin-right: 500px;
    background: white;
    color: black;
  }
  #service{
      font-size: 30px;
      color: orangered;
      font-weight: bold;
      font-family: cursive;
  }
```

Not Found Component

It is the page for not found component. When an url not in app.module.ts is entered. It'll direct to Not Found Component.



Not-Found-Component.ts – It is the not found component.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.css']
})
export class NotFoundComponent implements OnInit {

  constructor() { }

  ngOnInit() {
```

```
  }
}
```

Not-Found-Component.html – It is the html associated with the Not Found Component.

```
<img id="image" src="assets/pageNotFound.jpg" alt="Page not Found" height="550">
```

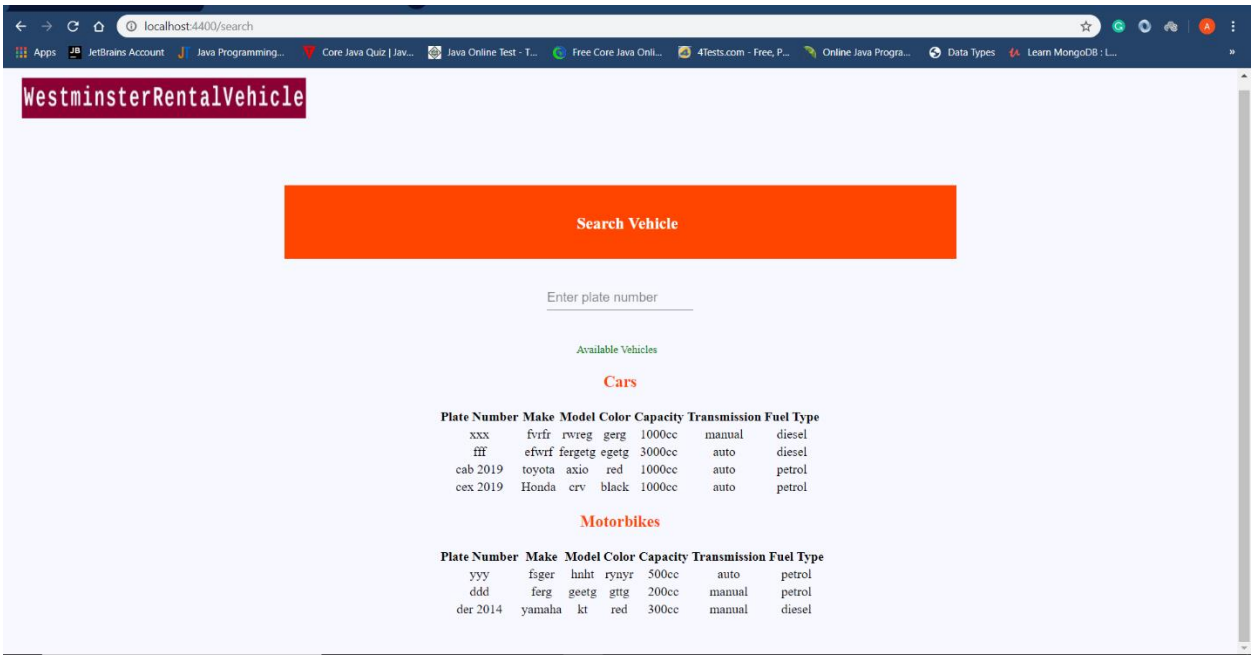
Not-Found-Component.css – It is the css associated with the Not Found Component.

```
#image{
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 50%;
}
```

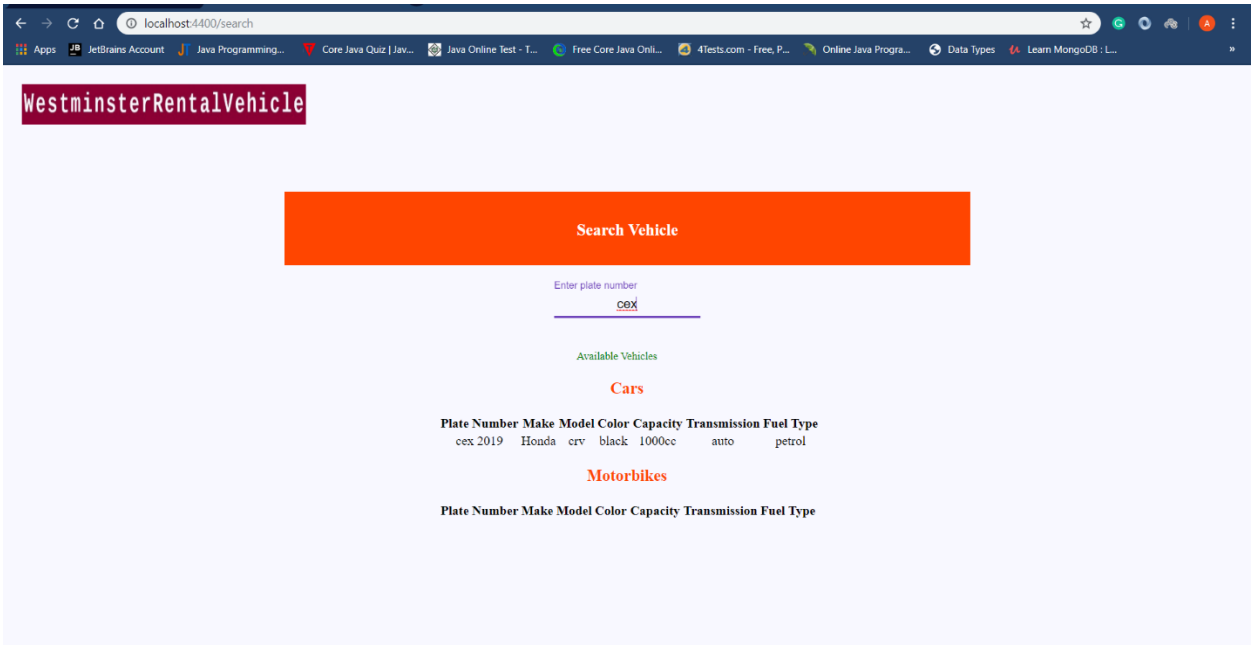
Search Component

According to the use case diagram the Manager can do a functionality of checking the availability of vehicles. In the console application the manager has an option to search vehicle. Once selected it the manager will be directed to the Search Component. In it he/she can search for vehicle by the plate number.

View of the Search Component;



Finding vehicle by the plate number;



Search-Result.Component.ts – It is the Search Component

```
import { Component, OnInit } from '@angular/core';
import {MainServiceService} from "../mainService/main-service.service";
import {Car} from "../models/Car";
import {Motorbike} from "../models/Motorbike";
import {CarService} from "../mainService/Car/car.service";
import {MotorbikeService} from "../mainService/Motorbike/motorbike.service";

@Component({
  selector: 'app-search-results',
  templateUrl: './search-results.component.html',
  styleUrls: ['./search-results.component.css']
})
export class SearchResultsComponent implements OnInit {
  cars:Car[];
  motorbikes:Motorbike[];
  plateNumber:String;

  constructor(private carService:CarService,private
motorbikeServices?:MotorbikeService) { }

  ngOnInit() {
    this.getCars();
    this.getMotorbike();
  }

  getCars(){
    this.carService.getCars().subscribe(
      response => {
        console.log(response);
        this.cars = response;
      }
    );
```

```
  }

  getMotorbike(){
    this.motorbikeServices.getMotorbike().subscribe(
      response => {
        console.log(response);
        this.motorbikes = response;
      }
    );
  }

}
```

Search-Result.Component.html – It is the html associated to Search Component

```
<body>
<div class="content" style="padding-left: 200px">
  <ul>
    <li><a routerLink="/search">Search Vehicle</a></li>
  </ul>
</div>
<br>
<div id="search">
  <mat-form-field>
    <input matInput  placeholder="Enter plate number" [(ngModel)]="plateNumber">
  </mat-form-field>
</div>
<br>
<small style="color: green;text-align: center;padding-left: 660px">Available
Vehicles</small>
<p>Cars</p>
<div style="text-align: center;margin-left: 490px;">
<table><thead></thead>
  <tr>
    <th>Plate Number</th>
    <th>Make</th>
    <th>Model</th>
    <th>Color</th>
    <th>Capacity</th>
    <th>Transmission</th>
    <th>Fuel Type</th>
  </tr>

  <tbody>
  <tr *ngFor="let car of cars | plateNumberCarFilter:plateNumber">
    <td>{{car.plateNumber}}</td>
    <td>{{car.make}}</td>
    <td>{{car.model}}</td>
    <td>{{car.color}}</td>
    <td>{{car.capacity}}</td>
    <td>{{car.transmission}}</td>
    <td>{{car.fuelType}}</td>
  </tr>
  </tbody>
```

```
</table>
</div>
<p>Motorbikes</p>
<div style="text-align: center;margin-left: 490px;">
<table>
  <thead></thead>
  <tr>
    <th>Plate Number</th>
    <th>Make</th>
    <th>Model</th>
    <th>Color</th>
    <th>Capacity</th>
    <th>Transmission</th>
    <th>Fuel Type</th>
  </tr>

  <tbody>
  <tr *ngFor="let motorbike of motorbikes | plateNumberMotorbikeFilter:plateNumber">
    <td>{{motorbike.plateNumber}}</td>
    <td>{{motorbike.make}}</td>
    <td>{{motorbike.model}}</td>
    <td>{{motorbike.color}}</td>
    <td>{{motorbike.capacity}}</td>
    <td>{{motorbike.transmission}}</td>
    <td>{{motorbike.fuelType}}</td>
  </tr>
  </tbody>
</table>
</div>
</body>
```

Search-Result.Component.css – It is the css associated to Search Component

```
p{
  text-align: center;
  color: orangered;
  font-size: 20px;
  font-weight: bold;
}
#searchBar{
  text-align: center;
  width: 50%;
  font-weight: bold;
  border-radius: 2px;
  border: 1px solid black;
  height: 30px;
}
#search{
  text-align: center;
  align-items: center;
}
#icon{
  color: black;
}
```

```
button{
  background: orangered;
  color: white;
  font-weight: bold;
  border: orangered;
  padding: 10px;
  border-radius: 5px;
  margin: 10px;
  cursor: pointer;
  width: 120px;
}
.buttons{
  text-align: right;
}
.content{
  background: orangered;
  padding: 20px;
  margin-left:300px ;
  margin-right: 300px;
  color: white;
  font-weight: bold;
  height: 50px;
}
.content ul li{
  list-style-type: none;
  float: left;
  width: 365px;
  text-align: center;
}
.content ul li a{
  display: block;
  color: white;
  font-weight: bold;
  text-decoration: none;
  font-size: 20px;
}
#vertical{
  border-left: 1px solid white;
  height: 50px;
  position:absolute;
  left: 50%;
}
input{
  text-align: center;
}
```

Filtering by plate number:

Plate-Number-Car-Filter.pipe.ts – Filtering plate number of cars.

```
import { Pipe, PipeTransform } from '@angular/core';
import {Car} from "../../../models/Car";

@Pipe({
```

```
  name: 'plateNumberCarFilter'
})
export class PlateNumberCarFilterPipe implements PipeTransform {

  transform(cars:Car[],plateNumber:string): Car[] {
    if(!cars || !plateNumber){
      return cars;
    }
    return cars.filter(car =>
car.plateNumber.toLowerCase().indexOf(plateNumber.toLowerCase()) !==-1);
  }

}
```

Plate-Number-Motorbike-Filter.pipe.ts – Filtering plate number of motorbikes.

```
import { Pipe, PipeTransform } from '@angular/core';
import {Motorbike} from "../../../models/Motorbike";

@Pipe({
  name: 'plateNumberMotorbikeFilter'
})
export class PlateNumberMotorbikeFilterPipe implements PipeTransform {

  transform(motorbike:Motorbike[],plateNumber:string): Motorbike[] {
    if(!motorbike || !plateNumber){
      return motorbike;
    }
    return motorbike.filter(motorbike =>
motorbike.plateNumber.toLowerCase().indexOf(plateNumber.toLowerCase()) !==-1);
  }

}
```

Services used are CarService.ts and MotorbikeService.ts [code written above].

Spring boot Part

The file that runs the spring boot;

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Configuration;

@SpringBootApplication
@Configuration
public class DemoApplication {

    public static void main(String[] args) {
```

```
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

VehicleController.java

This class has all the get and post request methods.

```
package com.example.demo.Controller;

import com.example.demo.Manager.Car;
import com.example.demo.Manager.Motorbike;
import com.example.demo.Manager.Vehicle;
import com.example.demo.Manager.WestminsterRentalVehicleManager;
import com.example.demo.Models.CarBooking;
import com.example.demo.Repository.CarRepo;
import com.example.demo.Repository.MotorbikeRepo;
import com.example.demo.Service.CarService;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class VehicleController {
//    //Creating Mongo Client
//    MongoClient mongoClient = new MongoClient("localhost",27017);
//
//    //Creating credentials
//    MongoCredential credential =
MongoCredential.createCredential("Akshaan","VehicleDatabase","akshaan123456".toCharAr
ray());
//
//
//    //Accessing Database
//    MongoDatabase database = mongoClient.getDatabase("VehicleDatabase"); //creates
the database
//
//
//
//    MongoCollection<Document> collection = database.getCollection("Bookings");
//    MongoCollection<Document> cars = database.getCollection("Cars");

    WestminsterRentalVehicleManager manager = new WestminsterRentalVehicleManager();

    @Autowired
    CarRepo carRepo;

    @Autowired
    MotorbikeRepo motorbikeRepo;
```

```
    @Autowired
    CarService carService;


//    @GetMapping("/vehicles")
//    public List<Vehicle> getVehicles(){
//        return vehicleRepo.findAll();
//    }

    @RequestMapping("/hi")
    public String getMessage(){
        return "Hi!";
    }

    @CrossOrigin(origins = "http://localhost:4400")
    @RequestMapping(value = "/cars",method = RequestMethod.GET,produces =
"application/json")
    public List<Car> getCars(){
        return manager.getCars();
    }

    @CrossOrigin(origins = "http://localhost:4400")
    @RequestMapping(value = "/motorbikes", method = RequestMethod.GET,produces =
"application/json")
    public List<Motorbike> getMotorbike(){
        return manager.getMotorbikes();
    }

    @CrossOrigin(origins = "http://localhost:4400")
    @RequestMapping(value = "/vehicles", method = RequestMethod.GET,produces =
"application/json")
    public List<Vehicle> getVehicles(){
        return manager.getVehicle();
    }

//    @CrossOrigin(origins = "http://localhost:4400")
//    @RequestMapping(value = "/cars/{plateNumber}",method =
RequestMethod.GET,produces = "application/json")
//    public Car getCarsOnPlateNumber(@PathVariable String plateNumber){
//        for (Car car: manager.getCars()) {
//            if(car.getPlateNumber().equalsIgnoreCase(plateNumber)){
//                return car;
//            }
//        }
//        return null;
//    }
//
//    @CrossOrigin(origins = "http://localhost:4400")
//    @RequestMapping(value = "/motorbikes/{plateNumber}",method =
RequestMethod.GET,produces = "application/json")
//    public Motorbike getMotorbikeOnPlateNumber(@PathVariable String plateNumber){
//        for (Motorbike motorbike: manager.getMotorbikes()) {
//            if(motorbike.getPlateNumber().equalsIgnoreCase(plateNumber)){
//                return motorbike;
```

```
//            }
//        }
//        return null;
//    }

//    @CrossOrigin(origins = "http://localhost:4400")
//    @RequestMapping(value = "/bookingcar/{car}/{email}/{nic}",method =
RequestMethod.POST,produces = "application/json")
//    public String bookCar(@PathVariable Car car, @PathVariable String email,
@PathVariable String nic, @RequestParam Date pickup, @RequestParam Date dropoff){
//        CarBooking carBooking =
carService.createCarBooking(car,email,nic,pickup,dropoff);
//        return carBooking.toString();
//    }
//    @RequestMapping(value = "/bookingmotorbike/{motorbike}/{email}/{nic}",method =
RequestMethod.POST,produces = "application/json")
//    public void bookMotorbike(@PathVariable Motorbike motorbike,@PathVariable
String email,@PathVariable String nic,@RequestParam Date pickup,@RequestParam Date
dropoff){
//
//    }

    @RequestMapping(value = "/value",method = RequestMethod.GET,produces =
"application/json")
    public String car(){
        return "{message}:datapassed";
    }

    @CrossOrigin(origins = "http://localhost:4400")
    @RequestMapping(value = "/bookingcar", method = RequestMethod.POST,produces =
"application/json")
    public String bookCar(@RequestBody CarBooking carBooking) throws
JsonProcessingException {
        Car car = getCar(carBooking.plateNumber);
//        Document document1 = new Document();
//        document1.append("car", car);
//        document1.append("car booking", carBooking);
//        MongoCollection<Document> collection = database.getCollection("Bookings");
//        collection.insertOne(document1);
        CarBooking carBooking1 =
carService.createBooking(carBooking.plateNumber,carBooking.email,carBooking.nic,carBo
oking.pickUpDate,carBooking.dropOffDate);
        ObjectWriter ow = new ObjectMapper().writer().withDefaultPrettyPrinter();
        String json = ow.writeValueAsString(carBooking1);
        return json;
    }

    @CrossOrigin(origins = "http://localhost:4400")
    @RequestMapping(value = "/bookingmotorbikes", method =
RequestMethod.POST,produces = "application/json")
    public void bookMotorbike(){

    }

    public Car getCar(String plateNumber){
```

```
        for (Car car: manager.getCars()) {
            if(car.getPlateNumber().equalsIgnoreCase(plateNumber)){
                return car;
            }
        }
        return null;
    }

    public Motorbike getMotorbike(String plateNumber){
        for (Motorbike motorbike: manager.getMotorbikes()) {
            if(motorbike.getPlateNumber().equalsIgnoreCase(plateNumber)){
                return motorbike;
            }
        }
        return null;
    }
}
```

Manager package

This package has classes that is like OOP classes above. But, according to the needs these files have been edited and makes non like the above OOP classes.

Below written are only the need class for spring boot. Others are not written.

Vehicle.java

```
package com.example.demo.Manager;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.web.bind.annotation.CrossOrigin;

import javax.persistence.Inheritance;
import java.util.Objects;

@CrossOrigin(origins = "http://localhost:4400")
public class Vehicle implements Comparable<Vehicle>{
    protected Object objectId;
    @Id
    protected String plateNumber;
    protected String make;
    protected String model;
    protected String color;
    protected String capacity;
    protected String transmission;
    protected String fuelType;
    protected int yearOfProduction;
    protected Schedule bookVehicle;
    protected Schedule pickUpDropOff;

    public Vehicle(){

    }
```

```
    public Vehicle(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction) {
        super();
        this.plateNumber = plateNumber;
        this.make = make;
        this.model = model;
        this.color = color;
        this.capacity = capacity;
        this.transmission = transmission;
        this.fuelType = fuelType;
        this.yearOfProduction = yearOfProduction;
    }

//    public Vehicle(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction, Schedule
bookVehicle) {
//        super();
//        this.plateNumber = plateNumber;
//        this.make = make;
//        this.model = model;
//        this.color = color;
//        this.capacity = capacity;
//        this.transmission = transmission;
//        this.fuelType = fuelType;
//        this.yearOfProduction = yearOfProduction;
//        this.bookVehicle = bookVehicle;
//    }
//
//    public Vehicle(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, Schedule pickUpDropOff ,int
yearOfProduction) {
//        super();
//        this.plateNumber = plateNumber;
//        this.make = make;
//        this.model = model;
//        this.color = color;
//        this.capacity = capacity;
//        this.transmission = transmission;
//        this.fuelType = fuelType;
//        this.yearOfProduction = yearOfProduction;
//        this.pickUpDropOff = pickUpDropOff;
//    }

    public String getPlateNumber() {
        return plateNumber;
    }


    public String getMake() {
        return make;
    }


    public String getColor() {
```

```
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }



    public String getCapacity() {
        return capacity;
    }


    public String getTransmission() {
        return transmission;
    }


    public String getFuelType() {
        return fuelType;
    }


    public int getYearOfProduction() {
        return yearOfProduction;
    }


    public Schedule getBookVehicle() {
        return bookVehicle;
    }

    public void setBookVehicle(Schedule bookVehicle) {
        this.bookVehicle = bookVehicle;
    }

    public Schedule getPickUpDropOff() {
        return pickUpDropOff;
    }

    public void setPickUpDropOff(Schedule pickUpDropOff) {
        this.pickUpDropOff = pickUpDropOff;
    }

    public String getModel() {
        return model;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Vehicle vehicle = (Vehicle) o;
        return yearOfProduction == vehicle.yearOfProduction &&
```

```
                Objects.equals(objectId, vehicle.objectId) &&
                plateNumber.equals(vehicle.plateNumber) &&
                make.equals(vehicle.make) &&
                model.equals(vehicle.model) &&
                color.equals(vehicle.color) &&
                capacity.equals(vehicle.capacity) &&
                transmission.equals(vehicle.transmission) &&
                fuelType.equals(vehicle.fuelType) &&
                bookVehicle.equals(vehicle.bookVehicle) &&
                pickUpDropOff.equals(vehicle.pickUpDropOff);
    }

    @Override
    public int hashCode() {
        return Objects.hash(objectId, plateNumber, make, model, color, capacity,
transmission, fuelType, yearOfProduction, bookVehicle, pickUpDropOff);
    }

    @Override
    public String toString() {
        return "Vehicle{" +
                "plateNumber='" + plateNumber + '\'' +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                ", capacity='" + capacity + '\'' +
                ", transmission='" + transmission + '\'' +
                ", fuelType='" + fuelType + '\'' +
                ", yearOfProduction=" + yearOfProduction +
                ", bookVehicle=" + bookVehicle +
                ", pickUpDropOff=" + pickUpDropOff +
                '}';
    }

    @Override
    public int compareTo(Vehicle o) {
        return this.getMake().compareTo(o.getMake());
    }
}
```

Car.java

```
package com.example.demo.Manager;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Persistent;
import org.springframework.data.annotation.TypeAlias;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.web.bind.annotation.CrossOrigin;

import java.util.Objects;
@CrossOrigin(origins = "http://localhost:4400")
@Document(collection = "Vehicles")
public class Car extends Vehicle {
```

```
    private int noOfDoors;
    private int noOfSeats;
    private String hybridStatus;

    public Car(){
        super();
    }

    public Car(String plateNumber, String make, String model, String color, String
capacity, String transmission, String fuelType, int yearOfProduction, int noOfDoors,
int noOfSeats,String hybridStatus) {
        super(plateNumber, make, model,color, capacity, transmission, fuelType,
yearOfProduction);
        this.noOfDoors = noOfDoors;
        this.noOfSeats = noOfSeats;
        this.hybridStatus = hybridStatus;
    }
//
//    public Car(String plateNumber, String make, String model, String color, String
capacity, String transmission, String fuelType, int yearOfProduction, Schedule
bookVehicle, int noOfDoors,int noOfSeats, String hybridStatus) {
//        super(plateNumber, make, model, color, capacity, transmission, fuelType,
yearOfProduction, bookVehicle);
//        this.noOfDoors = noOfDoors;
//        this.noOfSeats = noOfSeats;
//        this.hybridStatus = hybridStatus;
//    }
//
//    public Car(String plateNumber, String make, String model, String color, String
capacity, String transmission, String fuelType, Schedule pickUpDropOff, int
yearOfProduction, int noOfDoors,int noOfSeats, String hybridStatus) {
//        super(plateNumber, make, model, color, capacity, transmission, fuelType,
pickUpDropOff, yearOfProduction);
//        this.noOfDoors = noOfDoors;
//        this.noOfSeats = noOfSeats;
//        this.hybridStatus = hybridStatus;
//    }

    public int getNoOfDoors() {
        return noOfDoors;
    }


    public String getHybridStatus() {
        return hybridStatus;
    }

    public int getNoOfSeats() {
        return noOfSeats;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
```

```
        if (!super.equals(o)) return false;
        Car car = (Car) o;
        return noOfDoors == car.noOfDoors &&
                noOfSeats == car.noOfSeats &&
                hybridStatus.equals(car.hybridStatus);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), noOfDoors, noOfSeats, hybridStatus);
    }

    @Override
    public String toString() {
        return "Car{" +
                "plateNumber='" + plateNumber + '\'' +
                ", make='" + make + '\'' +
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                ", capacity='" + capacity + '\'' +
                ", transmission='" + transmission + '\'' +
                ", fuelType='" + fuelType + '\'' +
                ", yearOfProduction=" + yearOfProduction +
                ", bookVehicle=" + bookVehicle +
                ", pickUpDropOff=" + pickUpDropOff +
                ", noOfDoors=" + noOfDoors +
                ", noOfSeats=" + noOfSeats +
                ", hybridStatus='" + hybridStatus +
                '}';
    }
}
```

Motorbike.java

```
package com.example.demo.Manager;

import org.springframework.data.annotation.Persistent;
import org.springframework.data.annotation.TypeAlias;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.web.bind.annotation.CrossOrigin;

import java.util.Objects;
@CrossOrigin(origins = "http://localhost:4400")
@Document(collection = "Vehicles")
public class Motorbike extends Vehicle {
    protected String startType;
    protected String bikeType;

    public Motorbike(){
        super();
    }

    public Motorbike(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction, String
```

```
startType, String bikeType) {
        super(plateNumber, make, model, color, capacity, transmission, fuelType,
yearOfProduction);
        this.startType = startType;
        this.bikeType = bikeType;
    }

//    public Motorbike(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, int yearOfProduction, Schedule
bookVehicle, String startType, String bikeType) {
//        super(plateNumber, make, model, color, capacity, transmission, fuelType,
yearOfProduction, bookVehicle);
//        this.startType = startType;
//        this.bikeType = bikeType;
//    }
//
//    public Motorbike(String plateNumber, String make, String model, String color,
String capacity, String transmission, String fuelType, Schedule pickUpDropOff, int
yearOfProduction, String startType, String bikeType) {
//        super(plateNumber, make, model, color, capacity, transmission, fuelType,
pickUpDropOff, yearOfProduction);
//        this.startType = startType;
//        this.bikeType = bikeType;
//    }

    public String getStartType() {
        return startType;
    }

    public String getBikeType() {
        return bikeType;
    }



    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        Motorbike motorbike = (Motorbike) o;
        return startType.equals(motorbike.startType) &&
                bikeType.equals(motorbike.bikeType);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), startType, bikeType);
    }

    @Override
    public String toString() {
        return "Motorbike{" +
                ", plateNumber='" + plateNumber + '\'' +
                ", make='" + make + '\'' +
```

```
                ", model='" + model + '\'' +
                ", color='" + color + '\'' +
                ", capacity='" + capacity + '\'' +
                ", transmission='" + transmission + '\'' +
                ", fuelType='" + fuelType + '\'' +
                ", yearOfProduction=" + yearOfProduction +
                ", bookVehicle=" + bookVehicle +
                ", pickUpDropOff=" + pickUpDropOff +
                ", startType='" + startType +
                ", bikeType='" + bikeType +
                '}';
    }
}
```

WestminsterRentalVehicleManager.java

```
package com.example.demo.Manager;

import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

import java.awt.*;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

public class WestminsterRentalVehicleManager implements RentalVehicleManager {
    public static int spaces = 50;
    public static List<Vehicle> vehicles = new ArrayList<>(); //vehicles that are
added contains in this list
    public static List<Vehicle> deletedVehicles = new ArrayList<>(); //vehicles that
are deleted contains in this list
    private static File file = new File("Vehicles.txt");

    MongoClient mongoClient = new MongoClient("localhost",27017);
    MongoCredential credential =
MongoCredential.createCredential("Akshaan","VehicleDatabase","akshaan123456".toCharAr
ray());
    //Accessing Database
    MongoDatabase database = mongoClient.getDatabase("VehicleDatabase");

    @Override
    public boolean addVehicle(Vehicle vehicle){
        if (vehicles.size()<spaces) {
            vehicles.add(vehicle);
            spaces-=1;
            System.out.println(vehicle+" addition successful!");
            System.out.println("No of spaces used: "+getSpaces());
        }else {
```

```
            System.out.println("No space!");
        }
        return true;
    }

    @Override
    public boolean deleteVehicle(Vehicle vehicle) {
        deletedVehicles.add(vehicle);
        vehicles.remove(vehicle);
        spaces+=1;
        System.out.println(vehicle+" deletion successful!");
        System.out.println("No of available spaces: "+getSpaces());
        return true;
    }

    @Override
    public boolean listOfVehicle() {
        System.out.println(getVehicles());
        return true;
    }

    @Override
    public boolean save() {
        try {
            FileWriter fileWriter = new FileWriter(file);
            BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
            bufferedWriter.write("");
            bufferedWriter.newLine();
            bufferedWriter.write("-------------------------------------------------
---------------Vehicles-----------------------------------------------------------
----");
            bufferedWriter.newLine();
            bufferedWriter.write("");
            bufferedWriter.newLine();
            bufferedWriter.write("");
            bufferedWriter.newLine();
            for (int i = 0; i < getVehicles().size(); i++) {
                bufferedWriter.write((i+1)+") "+getVehicles().get(i));
                bufferedWriter.newLine();
                bufferedWriter.write("");
                bufferedWriter.newLine();
            }
            bufferedWriter.close();
        }catch (Exception e){
            System.out.println("File upload crashed!");
        }
        System.out.println("Saved successfully!");
        return true;
    }

    @Override
    public boolean bookVehicle(Vehicle vehicle) {
        return false;
    }
```

```
@Override
public boolean checkAvailability(Vehicle vehicle) {
    return false;
}

@Override
public void sortVehicles() {

}

@Override
public void generateReport() {
    try {
        if(getVehicles().size()==0){
            System.out.println("No vehicles!. Therefore, none to be shown...");
        }else {
            Desktop desktop = Desktop.getDesktop();
            if (!Desktop.isDesktopSupported()) {
                System.out.println("Not Supported!");
            }
            if (file.exists()) {
                desktop.open(file);
            } else {
                System.out.println("File cannot be found!");
            }
        }
    }catch (Exception e){
        System.out.println("File opening failed!");
    }
}

public static int getSpaces() {
    return spaces;
}

public static void setSpaces(int spaces) {
    WestminsterRentalVehicleManager.spaces = spaces;
}

public List<Vehicle> getVehicles() {
    return vehicles;
}

public static void setVehicles(List<Vehicle> vehicles) {
    WestminsterRentalVehicleManager.vehicles = vehicles;
}


public static List<Vehicle> getRDeletedVehicles() {
    return deletedVehicles;
}

public static void setDeletedVehicles(List<Vehicle> rentedVehicles) {
    WestminsterRentalVehicleManager.deletedVehicles = rentedVehicles;
}
```

```
    public ArrayList<Car> getCars(){
        ArrayList<Car> carList = new ArrayList<>();
        MongoCollection<Document> cars = database.getCollection("Cars");
        for (Document row:cars.find()) {
            Car car = new Car(
                    (String) row.get("plateNumber"),
                    (String) row.get("make"),
                    (String) row.get("model"),
                    (String) row.get("color"),
                    (String) row.get("capacity"),
                    (String) row.get("transmission"),
                    (String) row.get("fuelType"),
                    (int) row.get("yearOfProduction"),
                    (int) row.get("noOfDoors"),
                    (int) row.get("noOfSeats"),
                    (String) row.get("hybridStatus")
            );
            carList.add(car);
        }
        return carList;
    }
    public ArrayList<Motorbike> getMotorbikes(){
        ArrayList<Motorbike> motorbikeList = new ArrayList<>();
        MongoCollection<Document> cars = database.getCollection("Motorbikes");
        for (Document row:cars.find()) {
            Motorbike motorbike = new Motorbike(
                    (String) row.get("plateNumber"),
                    (String) row.get("make"),
                    (String) row.get("model"),
                    (String) row.get("color"),
                    (String) row.get("capacity"),
                    (String) row.get("transmission"),
                    (String) row.get("fuelType"),
                    (int) row.get("yearOfProduction"),
                    (String) row.get("startType"),
                    (String) row.get("bikeType")
            );
            motorbikeList.add(motorbike);
        }
        return motorbikeList;
    }
    public ArrayList<Vehicle> getVehicle(){
        ArrayList<Vehicle> vehicleList = new ArrayList<>();
        MongoCollection<Document> vehicleDatabase =
database.getCollection("Vehicles");
        for (Document row:vehicleDatabase.find()) {
            Vehicle vehicle = new Vehicle(
                    (String) row.get("plateNumber"),
                    (String) row.get("make"),
                    (String) row.get("model"),
                    (String) row.get("color"),
                    (String) row.get("capacity"),
                    (String) row.get("transmission"),
                    (String) row.get("fuelType"),
```

```
                    (int) row.get("yearOfProduction")
            );
            vehicleList.add(vehicle);
        }
        return vehicleList;
    }
}

class MakeComparator implements Comparator<Vehicle>{
    @Override
    public int compare(Vehicle o1, Vehicle o2) {
        return o1.getMake().compareTo(o2.getMake());
    }
}
```

Bookings

CarBookings.java

```
package com.example.demo.Models;
import com.example.demo.Manager.Car;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.Date;
import java.util.Objects;
@Document(collection = "Bookings")
public class CarBooking {
    public String plateNumber;
    public String email;
    public String nic;
    public Date pickUpDate;
    public Date dropOffDate;

    public CarBooking(String plateNumber, String email, String nic, Date pickUpDate,
Date dropOffDate) {
        this.plateNumber = plateNumber;
        this.email = email;
        this.nic = nic;
        this.pickUpDate = pickUpDate;
        this.dropOffDate = dropOffDate;
    }

    public CarBooking(Car car, CarBooking carBooking) {
    }

    public CarBooking() {
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        CarBooking that = (CarBooking) o;
        return plateNumber.equals(that.plateNumber) &&
                email.equals(that.email) &&
                nic.equals(that.nic) &&
```

```
                pickUpDate.equals(that.pickUpDate) &&
                dropOffDate.equals(that.dropOffDate);
    }

    @Override
    public int hashCode() {
        return Objects.hash(plateNumber, email, nic, pickUpDate, dropOffDate);
    }

    @Override
    public String toString() {
        return "CarBooking{" +
                "plateNumber='" + plateNumber + '\'' +
                ", email='" + email + '\'' +
                ", nic='" + nic + '\'' +
                ", pickUpDate=" + pickUpDate +
                ", dropOffDate=" + dropOffDate +
                '}';
    }
}
```

MotorbikeBookings.java

```
package com.example.demo.Models;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import java.util.Date;
import java.util.Objects;

@Document(collection = "Bookings")
public class MotorbikeBooking {
    public String plateNumber;
    public String email;
    public String nic;
    public Date pickUpDate;
    public Date dropOffDate;

    public MotorbikeBooking() {
    }

    public MotorbikeBooking(String plateNumber, String email, String nic, Date
pickUpDate, Date dropOffDate) {
        this.plateNumber = plateNumber;
        this.email = email;
        this.nic = nic;
        this.pickUpDate = pickUpDate;
        this.dropOffDate = dropOffDate;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        MotorbikeBooking that = (MotorbikeBooking) o;
        return plateNumber.equals(that.plateNumber) &&
```

```
                email.equals(that.email) &&
                nic.equals(that.nic) &&
                pickUpDate.equals(that.pickUpDate) &&
                dropOffDate.equals(that.dropOffDate);
    }

    @Override
    public int hashCode() {
        return Objects.hash(plateNumber, email, nic, pickUpDate, dropOffDate);
    }

    @Override
    public String toString() {
        return "MotorbikeBooking{" +
                "plateNumber='" + plateNumber + '\'' +
                ", email='" + email + '\'' +
                ", nic='" + nic + '\'' +
                ", pickUpDate=" + pickUpDate +
                ", dropOffDate=" + dropOffDate +
                '}';
    }
}
```

Repository

CarBookingRepository.java

```
package com.example.demo.Repository;

import com.example.demo.Models.CarBooking;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CarBookingRepo extends MongoRepository<CarBooking,String> {
    public CarBooking findByNic(String nic);
    public List<CarBooking> findByEmail(String email);
}
```

MotorbikeBookingRepository.java

```
package com.example.demo.Repository;


import com.example.demo.Models.MotorbikeBooking;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface MotorbikeBookingRepo extends
```

```
MongoRepository<MotorbikeBooking,String> {
    public MotorbikeBooking findByNic(String nic);
    public List<MotorbikeBooking> findByEmail(String email);
}
```

Services

CarServices.java

```
package com.example.demo.Service;

import com.example.demo.Manager.Car;
import com.example.demo.Models.CarBooking;
import com.example.demo.Repository.CarBookingRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.List;

@Service
public class CarService {
    @Autowired
    private CarBookingRepo carBookingRepo;

//    public CarBooking createCarBooking(Car car, String email, String nic, Date
pickUp, Date dropOff){
//        return carBookingRepo.save(new CarBooking(car,email,nic,pickUp,dropOff));
//    }

    public CarBooking createBooking(String plateNumber,String email, String nic, Date
pickUp, Date dropOff){
        return carBookingRepo.save(new
CarBooking(plateNumber,email,nic,pickUp,dropOff));
    }

    public List<CarBooking> getAllCarBooking(){
       return carBookingRepo.findAll();
    }
}
```

MotorbikeServices.java

```
package com.example.demo.Service;

import com.example.demo.Manager.Motorbike;
import com.example.demo.Models.MotorbikeBooking;
import com.example.demo.Repository.MotorbikeRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Date;

@Service
```

```
public class MotorbikeService {
    @Autowired
    private MotorbikeRepo motorbikeRepo;

//    public MotorbikeBooking createMotorbikeBooking(Motorbike motorbike,String
email,String nic, Date pickUp, Date dropOff){
//        return motorbikeRepo.save(new
MotorbikeBooking(motorbike,email,nic,pickUp,dropOff));
//    }
}
```

Testing

Test plan and Test Cases

Variable		Input	Expected	Test case status according to Junit
Option **File names:** Option.java and OptionTest.java		Any string value	Please reenter	For input of string, double and Boolean it prints "It's not a int" and the test case fails.
		Any number value	If number>=1 and number<=7: go to relevant option. Else: Please reenter	For inputs 1,2,3,4,5,6,7 it prints "Test case passed". For other inputs test case fails.
Add Vehicle (option==1)				
Type **File names:** Type.java and TypeTest.java		Any string value	Please reenter	For input of string, double and Boolean it prints "It's not a int" and the test case fails.
		Any number value	If number not equal to 1 or 2 Please reenter. Else go to relevant option	For inputs 1,2,3,4,5,6,7 it prints "Test case passed". For other inputs test case fails.
Car details (type==1) **File names:** Manager.java and Manager Test.java	Plate Number	Any string value	If length of plate number not equal to 6,7 or 8 Please reenter. Else go to other option.	For input of "cab 2019" test case passed. For inputs where length of the string neither 8,7 nor 6 failed.
	Make	Any value	Go to other option	Test case passed for any input.
	Model	Any value	Go to other option	Test case passed for any input.
	Color	Any value	Go to other option	Test case passed for any input.
	Capacity	Any value	If input doesn't contain cc, please reenter. Else go to other option.	When entered 1000cc test case passed. But, not for 1000.
	Transmission	Any value	If input not equal to auto or manual, please reenter. Else go to other option.	When entered auto or manual Test case passed. But, not for other inputs.

	Fuel type	Any value	If input not equal to petrol or diesel, please reenter. Else go to other option.	When entered petrol or diesel Test case passed. But, not for other inputs.
	Year of Production	Any string value	Please reenter	For input of string, double and Boolean the test case fails.
		Any number value	If year of production not less than or equal to 2019 Please reenter. Else go to other option.	For year/s above 2019 test case fails. Otherwise, the test case passes for years below and equal to 2019.
	No of doors	Any string value	Please reenter	For input of string, double and Boolean the test case fails.
		Any number value	If no of doors not > 1 or <6 Please reenter. Else go to other option.	For input >1 and <6 test case pass. But for others the test case fails.
	No of seats	Any string value	Please reenter	For input of string, double and Boolean the test case fails.
		Any number value	If no of seats not >= 2 or <10 Please reenter. Else go to other option.	For input >=2 and <10 test case pass. But for others the test case fails.
	Hybrid status	Any value	If hybrid status not equal to hybrid or nonhybrid Please reenter. Else go to other option.	When entered hybrid or nonhybrid Test case passed. But, not for other inputs.
Motorbike details (type==2) **File names:** Manager.java and Manager Test.java	Plate Number	Any value	If length of plate number not equal to 6,7 or 8 Please reenter. Else go to other option.	For input of "df 9345" test case passed. For inputs where length of the string neither 8,7 nor 6 failed.
	Make	Any value	Go to other option	Test case passed for any input.
	Model	Any value	Go to other option	Test case passed for any input.
	Color	Any value	Go to other option	Test case passed for any input.
	Capacity	Any value	If input doesn't contain cc, please reenter. Else go to other option.	When entered 500cc test case passed. But, not for 500.
	Transmission	Any value	If input not equal to auto or manual, please reenter. Else go to other option.	When entered auto or manual Test case passed. But, not for other inputs.
	Fuel type	Any value	If input not equal to petrol or diesel, please reenter. Else go to other option.	When entered petrol or diesel Test case passed. But, not for other inputs.
	Year of production	Any string value	Please reenter	For input of string, double and Boolean the test case fails.

		Any number value	If year of production not less than or equal to 2019 Please reenter. Else go to other option.	For year/s above 2019 test case fails. Otherwise, the test case passes for years below and equal to 2019.
	Start type	Any value	If start type not equal to key or push, please reenter. Else go to other option.	When entered key or push Test case passed. But, not for other inputs.
	Bike type	Any value	If bike type not equal to scooty or normal, please reenter. Else go to other option.	When entered scooty or normal Test case passed. But, not for other inputs.

Testing Code

Option.java

```
package testOptions;
import java.util.*;

public class Option {
	public int optionhasInt(Scanner input){
	    if(input.hasNextInt()) {
	        int option = input.nextInt();
	      return option;
	    }else {
	       System.err.println("It's not a int");
	    }
	    return 0;
    }
}
```

Type.java

```
package testOptions;

import java.util.Scanner;

public class Type {
	public int typehasInt(Scanner input){
	    if(input.hasNextInt()) {
	        int option = input.nextInt();
	      return option;
	    }else {
	       System.err.println("It's not a int");
	    }
	    return 0;
    }
}
```

ManagerWork.java

```
package testOptions;

public class ManagerWork {
	public String plateNumber(String plateNumber){
	    return plateNumber;
    }

	public String capacity(String capacity) {
		return capacity;
	}

	public String transmission(String transmission) {
		return transmission;
	}

	public String fuelType(String fuelType) {
		return fuelType;
	}

	public int yearOfProduction(int yearOfProduction) {
		return yearOfProduction;
	}

	public int noOfDoors(int noOfDoors) {
		return noOfDoors;
	}

	public int noOfSeats(int noOfSeats) {
		return noOfSeats;
	}

	public String hybridStatus(String hybridStatus) {
		return hybridStatus;
	}

	public String startType(String startType) {
		return startType;
	}

	public String bikeType(String bikeType) {
		return bikeType;
	}
}
```

OptionTest.java

```
package testOptions;


import static org.junit.jupiter.api.Assertions.*;


import org.junit.jupiter.api.BeforeEach;
```

```
import org.junit.jupiter.api.Test;

import java.util.Scanner;

public class OptionTest {
	private Option option;

	@BeforeEach
	void initEach() {
		option = new Option();
	}

	@Test
	public void testingOption() {
		System.out.println("Enter option: ");
		Scanner input = new Scanner(System.in);
		int optionNum = option.optionhasInt(input);
		if (optionNum>=1 && optionNum<=7){
			System.out.println("Test case passed");
		}else {
			fail("Test case failed");
		}
	}

}
```

TypeTest.java

```
package testOptions;

import static org.junit.jupiter.api.Assertions.*;
```

```
import java.util.Scanner;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class TypeTest {

	private Type type;

	@BeforeEach
	void initEach() {
		type = new Type();
	}

	@Test
	public void testingOption() {
		System.out.println("Enter option: ");
		Scanner input = new Scanner(System.in);
		int optionNum = type.typehasInt(input);
		if (optionNum==1 || optionNum==2){
			System.out.println("Test case passed");
		}else {
			fail("Test case failed");
		}
	}

}
```

ManagerWorkTest.java

```
package testOptions;
```

```
import static org.junit.jupiter.api.Assertions.*;

import java.util.Scanner;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class ManagerWorkTest {

	private ManagerWork managerWork;

	@BeforeEach
	void initEach() {
		managerWork = new ManagerWork();
	}

	@Test
	void plateNumbertest() {
		System.out.println("Enter plateNumber: ");
		Scanner input = new Scanner(System.in);
		String string = input.nextLine();
		String plateNumber = managerWork.plateNumber(string);
		if (!(plateNumber.length()==8 || plateNumber.length()==7|| plateNumber.length()==6)){
			fail("Test case failed");
      }else {
			System.out.println("Test case passed");
		}
		input.close();
	}
```

```
	@Test
	void capacitytest() {
		System.out.println("Enter capacity: ");
		Scanner input = new Scanner(System.in);
		String string = input.next();
		String capacity = managerWork.capacity(string);
		if (!capacity.contains("cc")) {
			fail("Test case failed");
	}else {
			System.out.println("Test case passed");
		}
		input.close();
	}

	@Test
	void transmissiontest() {
		System.out.println("Enter transmission: ");
		Scanner input = new Scanner(System.in);
		String string = input.next();
		String transmission = managerWork.transmission(string);
		if (!(transmission.equalsIgnoreCase("auto")||
(transmission.equalsIgnoreCase("manual")))) {
			fail("Test case failed");
	}else {
			System.out.println("Test case passed");
		}
		input.close();
	}

	@Test
```

```
    void fuelType() {
            System.out.println("Enter fuelType: ");
            Scanner input = new Scanner(System.in);
            String string = input.next();
            String fuelType = managerWork.fuelType(string);
            if (!(fuelType.equalsIgnoreCase("petrol")|| (fuelType.equalsIgnoreCase("diesel")))) {
                    fail("Test case failed");
}else {
                    System.out.println("Test case passed");
            }
            input.close();
    }


    @Test
    void yearOfProduction() {
            System.out.println("Enter yearOfProduction: ");
            Scanner input = new Scanner(System.in);
            int integer = input.nextInt();
            int yearOfProduction = managerWork.yearOfProduction(integer);
            if (!(yearOfProduction<=2019)) {
                    fail("Test case failed");
}else {
                    System.out.println("Test case passed");
            }
            input.close();
    }

    @Test
    void noOfDoors() {
```

```
            System.out.println("Enter noOfDoors: ");
            Scanner input = new Scanner(System.in);
            int integer = input.nextInt();
            int noOfDoors = managerWork.noOfDoors(integer);
            if (!(noOfDoors<6 && noOfDoors>1)) {
                  fail("Test case failed");
}else {
                  System.out.println("Test case passed");
            }
            input.close();
      }

      @Test
      void noOfSeats() {
            System.out.println("Enter noOfSeats: ");
            Scanner input = new Scanner(System.in);
            int integer = input.nextInt();
            int noOfSeats = managerWork.noOfSeats(integer);
            if (!(noOfSeats<10 && noOfSeats>=2)) {
                  fail("Test case failed");
}else {
                  System.out.println("Test case passed");
            }
            input.close();
      }

      @Test
      void hybridStatus() {
            System.out.println("Enter hybridStatus: ");
            Scanner input = new Scanner(System.in);
```

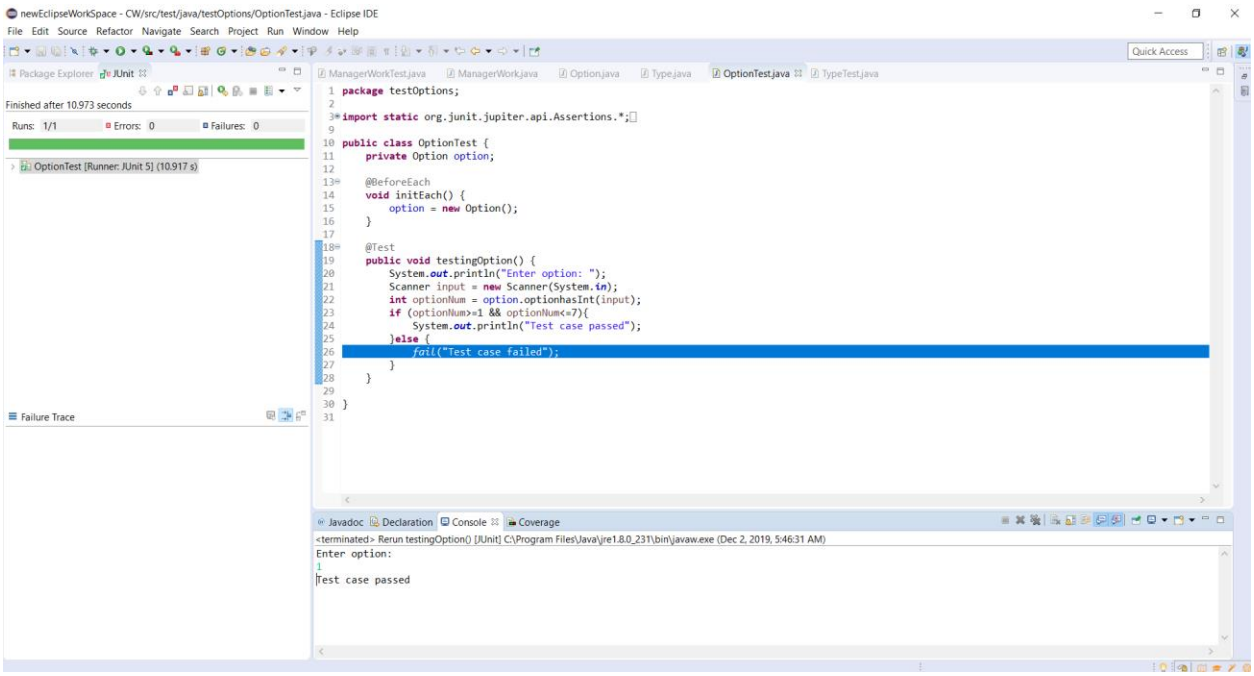
```
                String string = input.next();
                String hybridStatus = managerWork.hybridStatus(string);
                if (!((hybridStatus.equalsIgnoreCase("hybrid")||
(hybridStatus.equalsIgnoreCase("nonhybrid"))))) {
                        fail("Test case failed");
        }else {
                        System.out.println("Test case passed");
                }
                input.close();
        }

        @Test
        void startType() {
                System.out.println("Enter startType: ");
                Scanner input = new Scanner(System.in);
                String string = input.next();
                String startType = managerWork.startType(string);
                if (!((startType.equalsIgnoreCase("push")|| (startType.equalsIgnoreCase("key"))))) {
                        fail("Test case failed");
        }else {
                        System.out.println("Test case passed");
                }
                input.close();
        }

        @Test
        void bikeType() {
                System.out.println("Enter bikeType: ");
                Scanner input = new Scanner(System.in);
                String string = input.next();
```

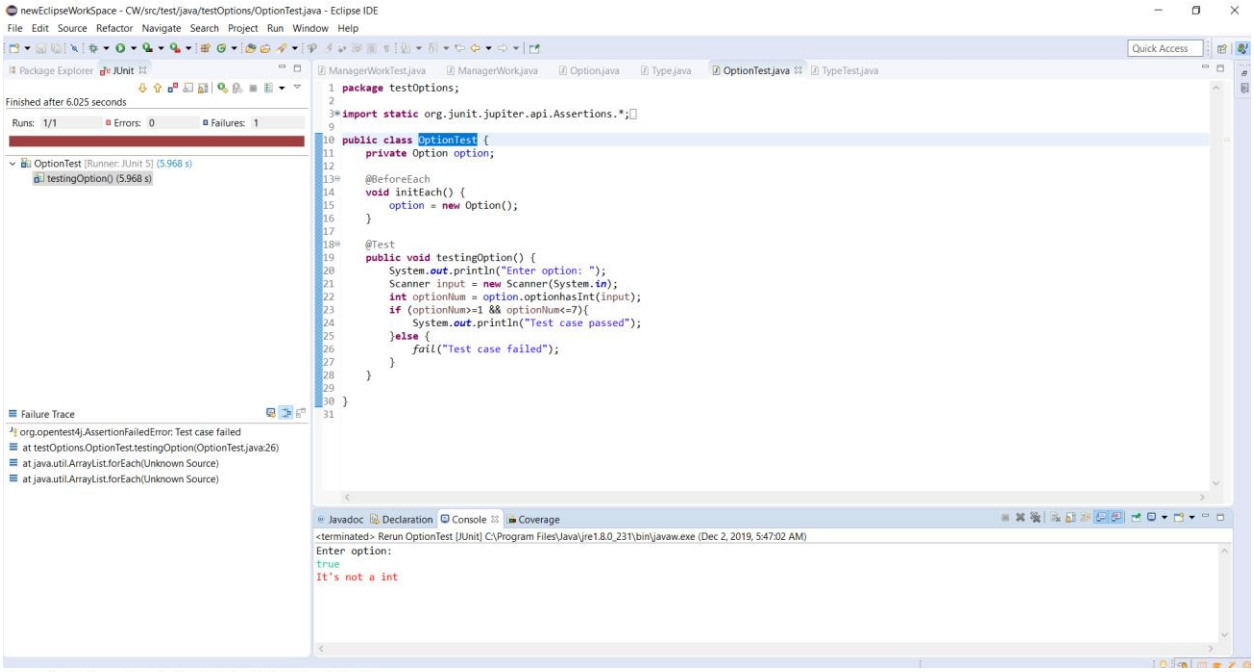
```
                String bikeType = managerWork.bikeType(string);
                if(!((bikeType.equalsIgnoreCase("scooty")|| (bikeType.equalsIgnoreCase("normal"))))){
                        fail("Test case failed");
        }else {
                        System.out.println("Test case passed");
                }
                input.close();
        }
}
```

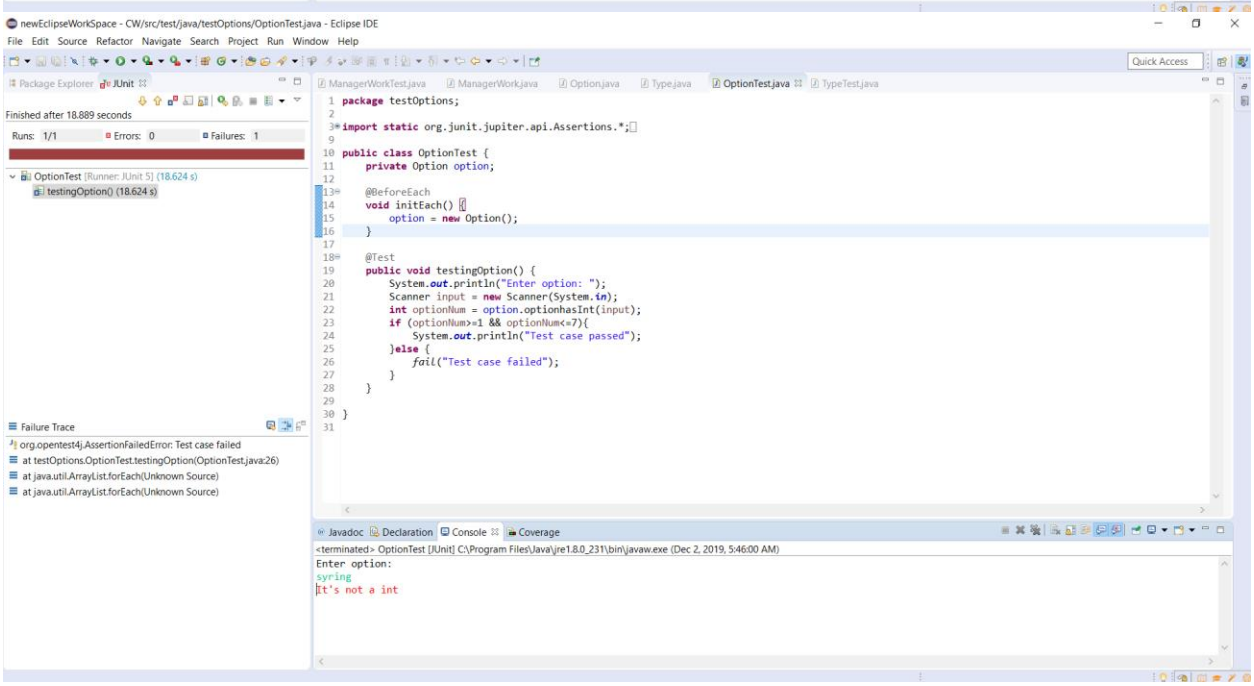
Testing Screenshots

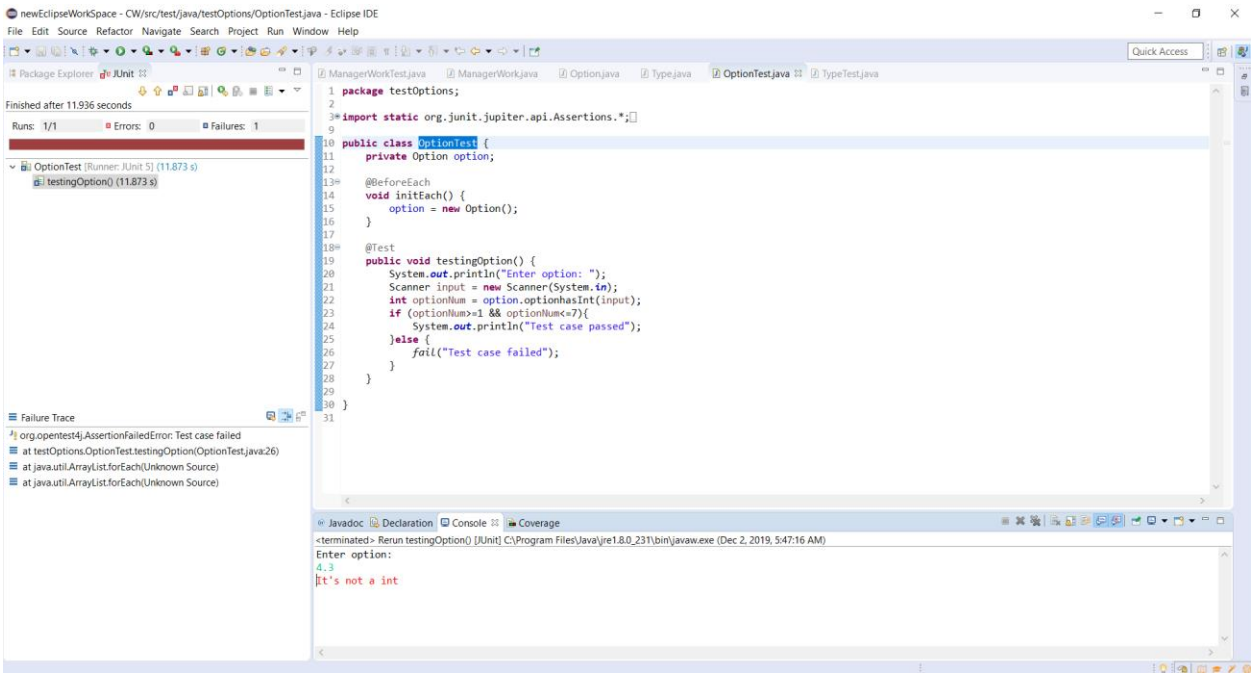
Below are screen shots for some test cases.

Option

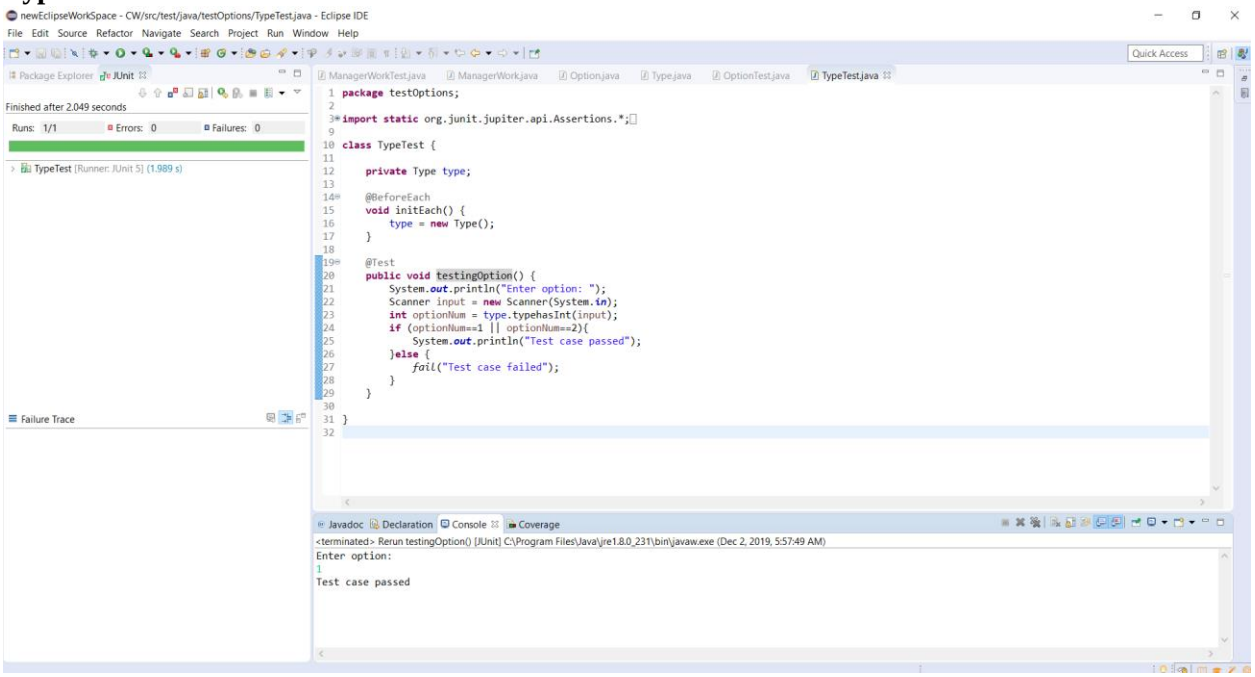


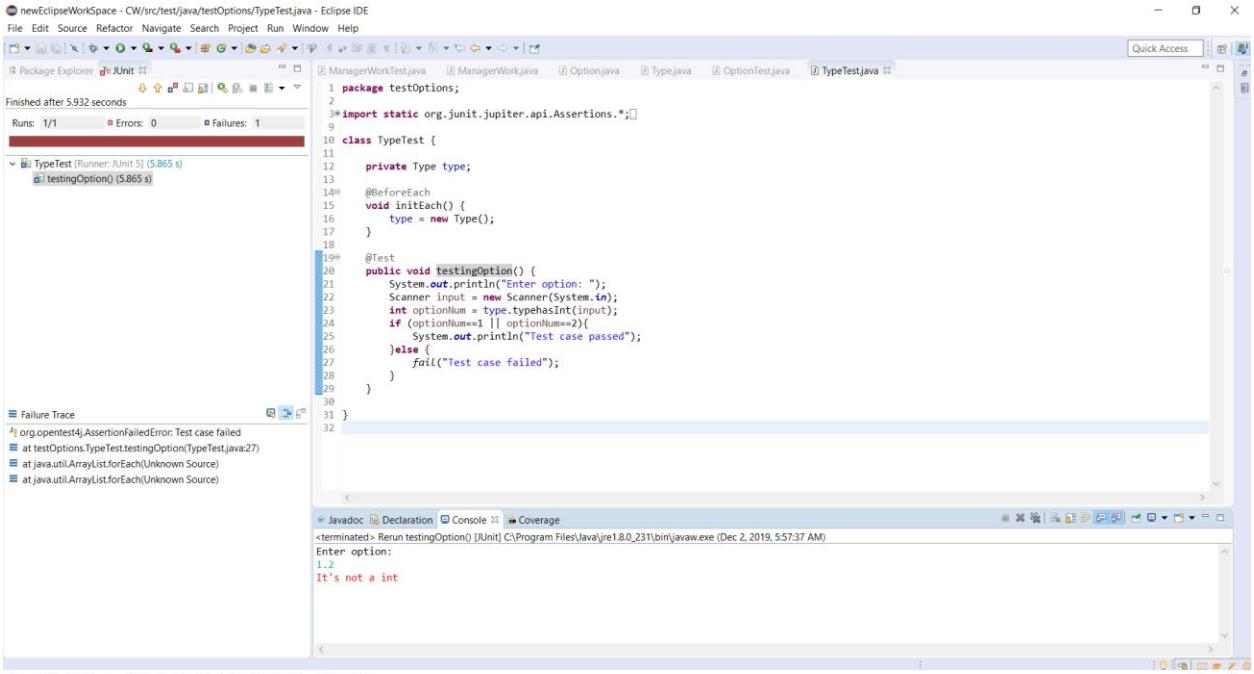

newEclipseWorkSpace - CW/src/test/java/testOptions/OptionTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 6.025 seconds
Runs: 1/1 Errors: 0 Failures: 1
OptionTest [Runner: JUnit 5] (5.968 s)
testingOption() (5.968 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.OptionTest.testingOption(OptionTest.java:26)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 public class OptionTest {
11 private Option option;
12
13 @BeforeEach
14 void initEach() {
15 option = new Option();
16 }
17
18 @Test
19 public void testingOption() {
20 System.out.println("Enter option: ");
21 Scanner input = new Scanner(System.in);
22 int optionNum = option.optionhasInt(input);
23 if (optionNum>=1 && optionNum<=7){
24 System.out.println("Test case passed");
25 }else {
26 fail("Test case failed");
27 }
28 }
29
30 }
31
Javadoc Declaration Console Coverage
<terminated> Rerun OptionTest [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:47:02 AM)
Enter option:
true
It's not a int

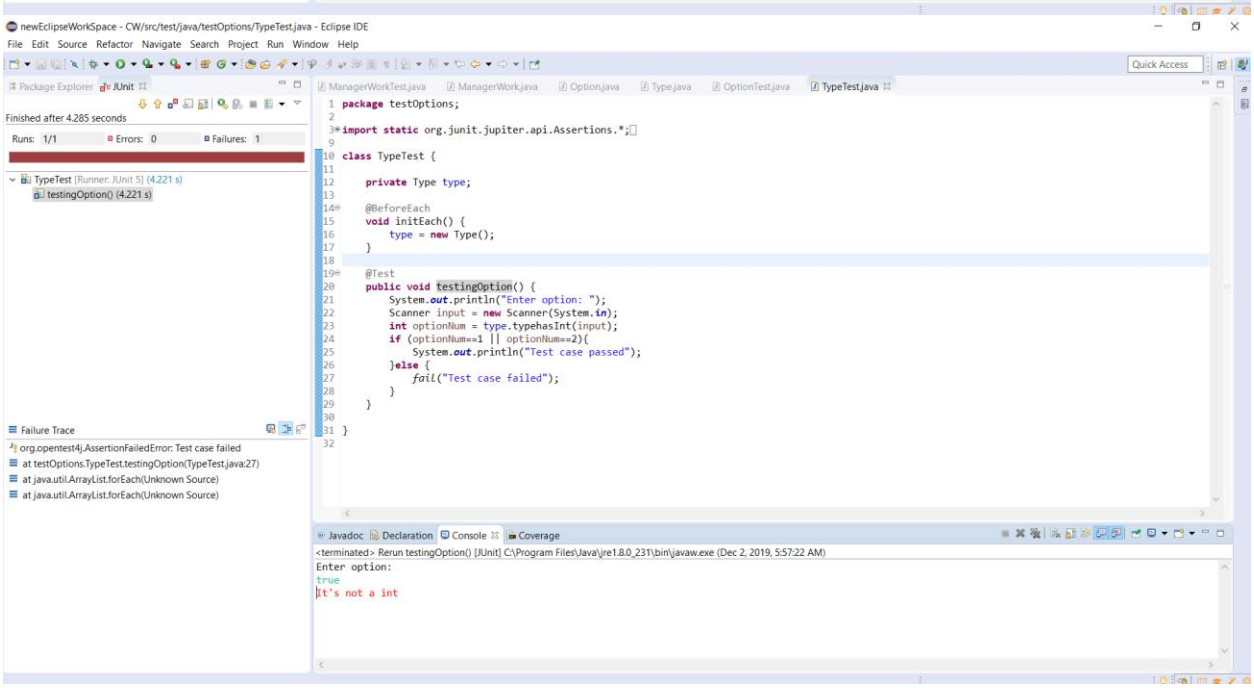

newEclipseWorkSpace - CW/src/test/java/testOptions/OptionTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 18.889 seconds
Runs: 1/1 Errors: 0 Failures: 1
OptionTest [Runner: JUnit 5] (18.624 s)
testingOption() (18.624 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.OptionTest.testingOption(OptionTest.java:26)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 public class OptionTest {
11 private Option option;
12
13 @BeforeEach
14 void initEach() {
15 option = new Option();
16 }
17
18 @Test
19 public void testingOption() {
20 System.out.println("Enter option: ");
21 Scanner input = new Scanner(System.in);
22 int optionNum = option.optionhasInt(input);
23 if (optionNum>=1 && optionNum<=7){
24 System.out.println("Test case passed");
25 }else {
26 fail("Test case failed");
27 }
28 }
29
30 }
31
Javadoc Declaration Console Coverage
<terminated> OptionTest [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:46:00 AM)
Enter option:
syring
It's not a int

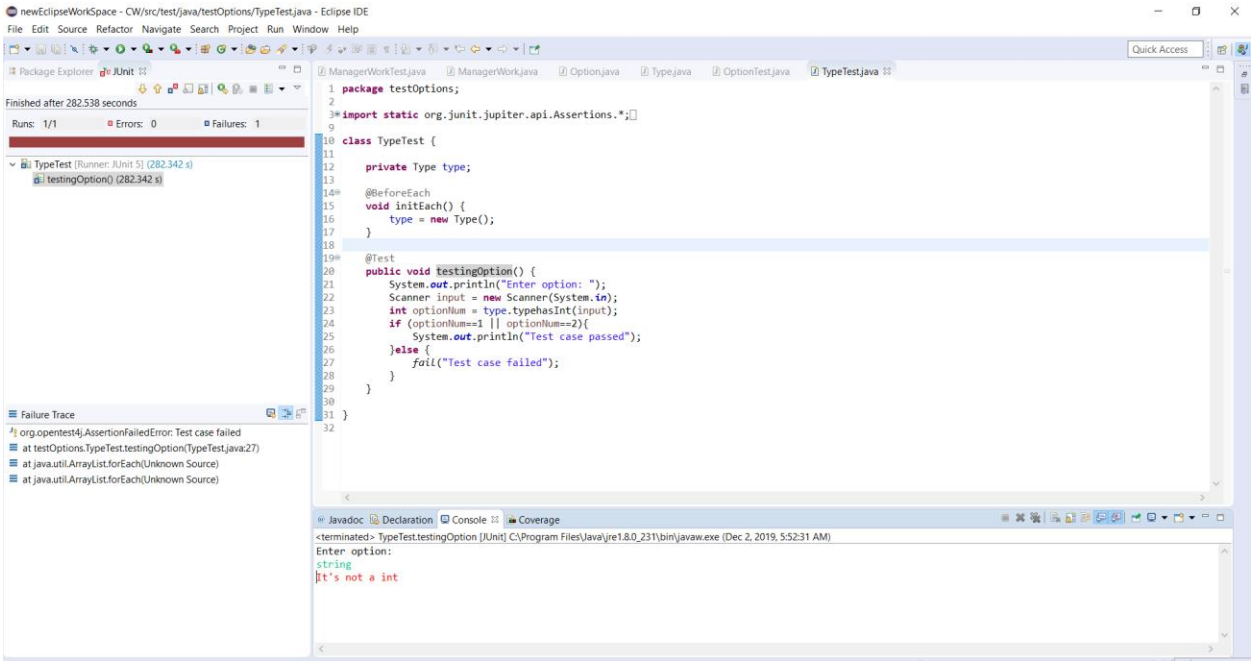

newEclipseWorkSpace - CW/src/test/java/testOptions/OptionTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 11.936 seconds
Runs: 1/1 Errors: 0 Failures: 1
OptionTest [Runner: JUnit 5] (11.873 s)
testingOption() (11.873 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.OptionTest.testingOption(OptionTest.java:26)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java
ManagerWork.java
Option.java
Type.java
OptionTest.java
TypeTest.java
1 package testOptions;
2
3* import static org.junit.jupiter.api.Assertions.*;
9
10 public class OptionTest {
11 private Option option;
12
13 @BeforeEach
14 void initEach() {
15 option = new Option();
16 }
17
18 @Test
19 public void testingOption() {
20 System.out.println("Enter option: ");
21 Scanner input = new Scanner(System.in);
22 int optionNum = option.optionhasInt(input);
23 if (optionNum>=1 && optionNum<=7){
24 System.out.println("Test case passed");
25 }else {
26 fail("Test case failed");
27 }
28 }
29
30 }
31
Javadoc Declaration Console Coverage
<terminated> Rerun testingOption() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:47:16 AM)
Enter option:
4.3
It's not a int

Type

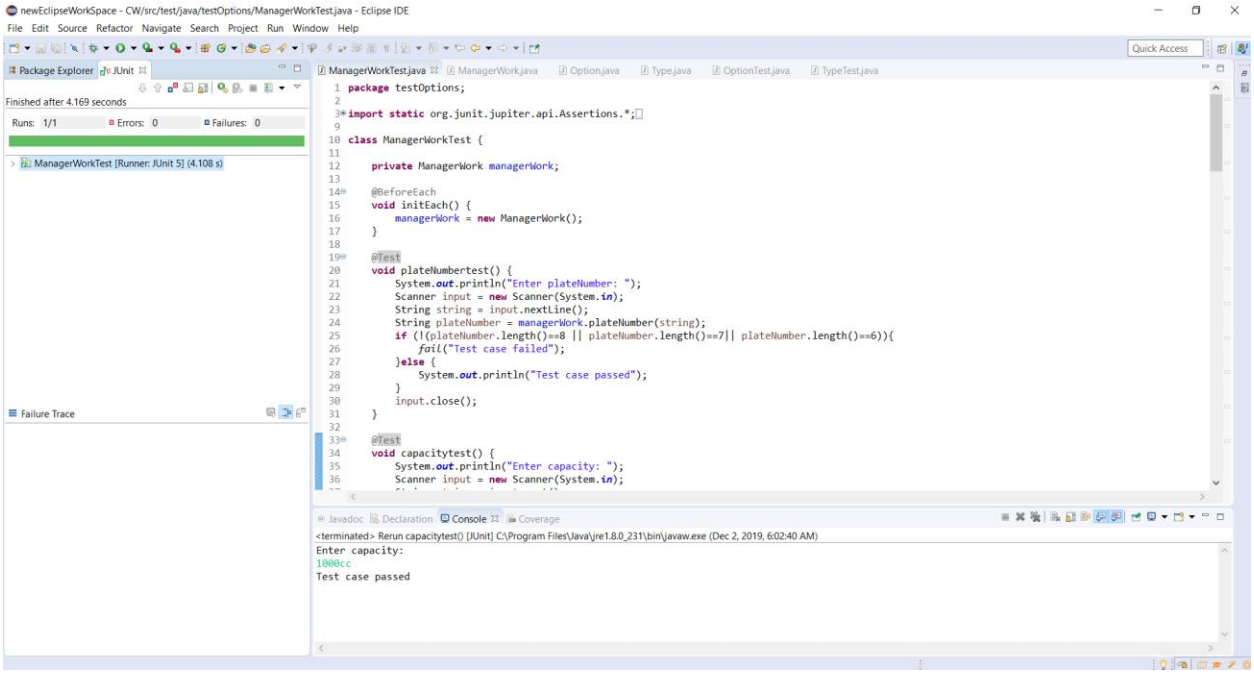

newEclipseWorkSpace - CW/src/test/java/testOptions/TypeTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 2.049 seconds
Runs: 1/1 Errors: 0 Failures: 0
TypeTest [Runner: JUnit 5] (1.989 s)
Failure Trace
ManagerWorkTest.java
ManagerWork.java
Option.java
Type.java
OptionTest.java
TypeTest.java
1 package testOptions;
2
3* import static org.junit.jupiter.api.Assertions.*;
9
10 class TypeTest {
11
12 private Type type;
13
14 @BeforeEach
15 void initEach() {
16 type = new Type();
17 }
18
19 @Test
20 public void testingOption() {
21 System.out.println("Enter option: ");
22 Scanner input = new Scanner(System.in);
23 int optionNum = type.typehasInt(input);
24 if (optionNum==1 || optionNum==2){
25 System.out.println("Test case passed");
26 }else {
27 fail("Test case failed");
28 }
29 }
30
31 }
32
Javadoc Declaration Console Coverage
<terminated> Rerun testingOption() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:57:49 AM)
Enter option:
1
Test case passed

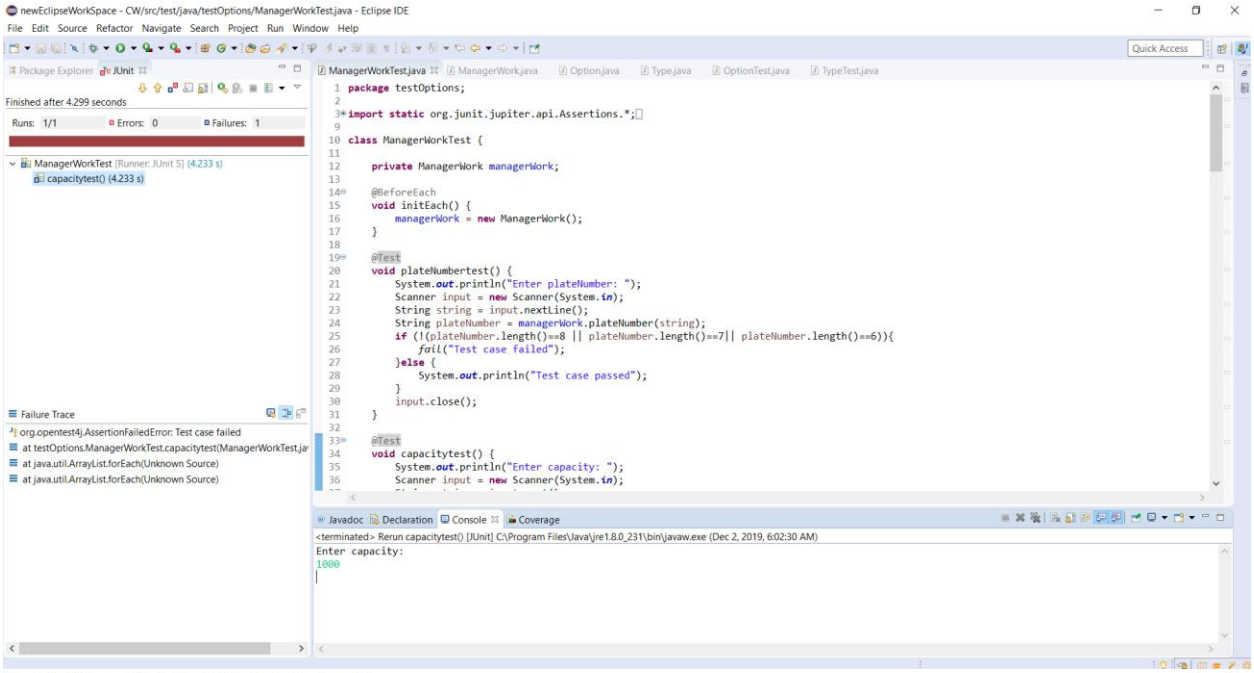

newEclipseWorkSpace - CW/src/test/java/testOptions/TypeTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 5.932 seconds
Runs: 1/1 Errors: 0 Failures: 1
TypeTest [Runner: JUnit 5] (5.865 s)
testingOption() (5.865 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.TypeTest.testingOption(TypeTest.java:27)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 class TypeTest {
11
12 private Type type;
13
14 @BeforeEach
15 void initEach() {
16 type = new Type();
17 }
18
19 @Test
20 public void testingOption() {
21 System.out.println("Enter option: ");
22 Scanner input = new Scanner(System.in);
23 int optionNum = type.typehasInt(input);
24 if (optionNum==1 || optionNum==2){
25 System.out.println("Test case passed");
26 }else {
27 fail("Test case failed");
28 }
29 }
30
31 }
32
Javadoc Declaration Console Coverage
<terminated> Rerun testingOption() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:57:37 AM)
Enter option:
1.2
It's not a int

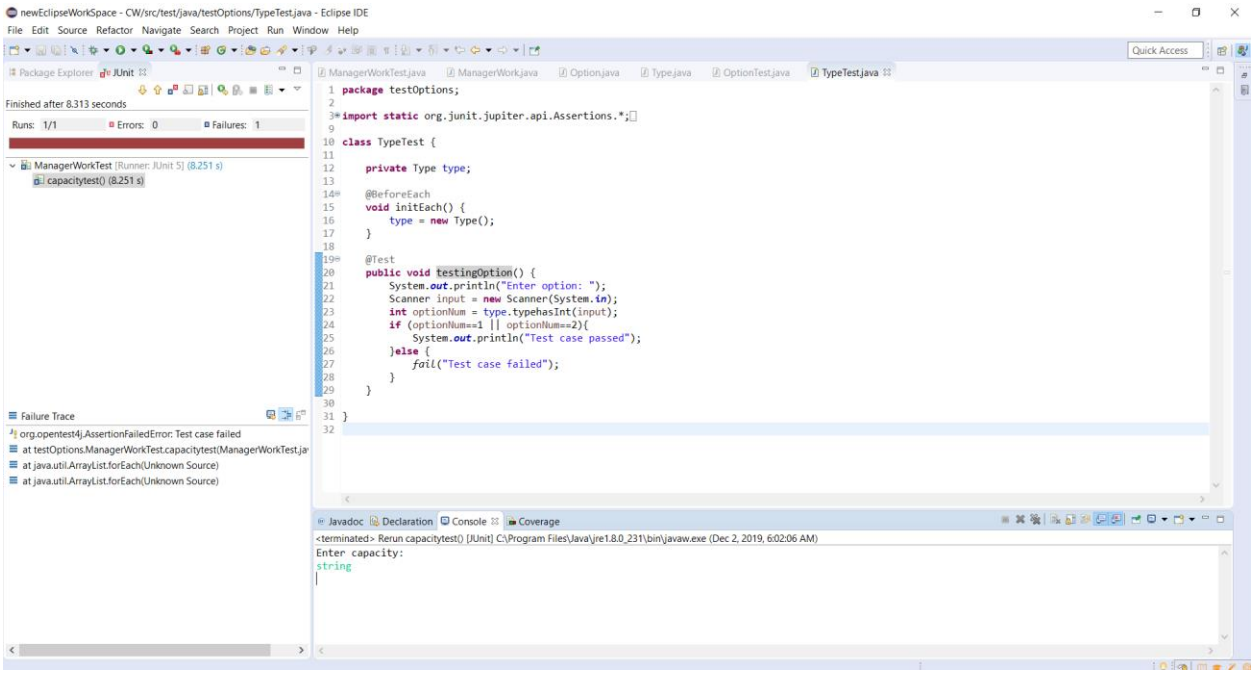

newEclipseWorkSpace - CW/src/test/java/testOptions/TypeTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 4.285 seconds
Runs: 1/1 Errors: 0 Failures: 1
TypeTest [Runner: JUnit 5] (4.221 s)
testingOption() (4.221 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.TypeTest.testingOption(TypeTest.java:27)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 class TypeTest {
11
12 private Type type;
13
14 @BeforeEach
15 void initEach() {
16 type = new Type();
17 }
18
19 @Test
20 public void testingOption() {
21 System.out.println("Enter option: ");
22 Scanner input = new Scanner(System.in);
23 int optionNum = type.typehasInt(input);
24 if (optionNum==1 || optionNum==2){
25 System.out.println("Test case passed");
26 }else {
27 fail("Test case failed");
28 }
29 }
30
31 }
32
Javadoc Declaration Console Coverage
<terminated> Rerun testingOption() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:57:22 AM)
Enter option:
true
It's not a int


newEclipseWorkSpace - CW/src/test/java/testOptions/TypeTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 282.538 seconds
Runs: 1/1 Errors: 0 Failures: 1
TypeTest [Runner: JUnit 5] (282.342 s)
testingOption() (282.342 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.TypeTest.testingOption(TypeTest.java:27)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 class TypeTest {
11
12 private Type type;
13
14 @BeforeEach
15 void initEach() {
16 type = new Type();
17 }
18
19 @Test
20 public void testingOption() {
21 System.out.println("Enter option: ");
22 Scanner input = new Scanner(System.in);
23 int optionNum = type.typehasInt(input);
24 if (optionNum==1 || optionNum==2){
25 System.out.println("Test case passed");
26 }else {
27 fail("Test case failed");
28 }
29 }
30
31 }
32
Javadoc Declaration Console Coverage
<terminated> TypeTest.testingOption [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 5:52:31 AM)
Enter option:
string
It's not a int

Capacity


newEclipseWorkSpace - CW/src/test/java/testOptions/ManagerWorkTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 4.169 seconds
Runs: 1/1 Errors: 0 Failures: 0
ManagerWorkTest [Runner: JUnit 5] (4.108 s)
Failure Trace
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 class ManagerWorkTest {
11
12 private ManagerWork managerWork;
13
14 @BeforeEach
15 void initEach() {
16 managerWork = new ManagerWork();
17 }
18
19 @Test
20 void plateNumbertest() {
21 System.out.println("Enter plateNumber: ");
22 Scanner input = new Scanner(System.in);
23 String string = input.nextLine();
24 String plateNumber = managerWork.plateNumber(string);
25 if (!(plateNumber.length()==8 || plateNumber.length()==7|| plateNumber.length()==6)){
26 fail("Test case failed");
27 }else {
28 System.out.println("Test case passed");
29 }
30 input.close();
31 }
32
33 @Test
34 void capacitytest() {
35 System.out.println("Enter capacity: ");
36 Scanner input = new Scanner(System.in);
Javadoc Declaration Console Coverage
<terminated> Rerun capacitytest() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 6:02:40 AM)
Enter capacity:
1000cc
Test case passed


newEclipseWorkSpace - CW/src/test/java/testOptions/ManagerWorkTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 4.299 seconds
Runs: 1/1 Errors: 0 Failures: 1
ManagerWorkTest [Runner: JUnit 5] (4.233 s)
capacitytest() (4.233 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.ManagerWorkTest.capacitytest(ManagerWorkTest.ja
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 class ManagerWorkTest {
11
12 private ManagerWork managerWork;
13
14 @BeforeEach
15 void initEach() {
16 managerWork = new ManagerWork();
17 }
18
19 @Test
20 void plateNumbertest() {
21 System.out.println("Enter plateNumber: ");
22 Scanner input = new Scanner(System.in);
23 String string = input.nextLine();
24 String plateNumber = managerWork.plateNumber(string);
25 if (!(plateNumber.length()==8 || plateNumber.length()==7|| plateNumber.length()==6)){
26 fail("Test case failed");
27 }else {
28 System.out.println("Test case passed");
29 }
30 input.close();
31 }
32
33 @Test
34 void capacitytest() {
35 System.out.println("Enter capacity: ");
36 Scanner input = new Scanner(System.in);
Javadoc Declaration Console Coverage
<terminated> Rerun capacitytest() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 6:02:30 AM)
Enter capacity:
1000


newEclipseWorkSpace - CW/src/test/java/testOptions/TypeTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer JUnit
Finished after 8.313 seconds
Runs: 1/1 Errors: 0 Failures: 1
ManagerWorkTest [Runner: JUnit 5] (8.251 s)
capacitytest() (8.251 s)
Failure Trace
org.opentest4j.AssertionFailedError: Test case failed
at testOptions.ManagerWorkTest.capacitytest(ManagerWorkTest.ja
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
ManagerWorkTest.java ManagerWork.java Option.java Type.java OptionTest.java TypeTest.java
1 package testOptions;
2
3 import static org.junit.jupiter.api.Assertions.*;
9
10 class TypeTest {
11
12 private Type type;
13
14 @BeforeEach
15 void initEach() {
16 type = new Type();
17 }
18
19 @Test
20 public void testingOption() {
21 System.out.println("Enter option: ");
22 Scanner input = new Scanner(System.in);
23 int optionNum = type.typehasInt(input);
24 if (optionNum==1 || optionNum==2){
25 System.out.println("Test case passed");
26 }else {
27 fail("Test case failed");
28 }
29 }
30
31 }
32
Javadoc Declaration Console Coverage
<terminated> Rerun capacitytest() [JUnit] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (Dec 2, 2019, 6:02:06 AM)
Enter capacity:
string