Coursework 1 (OOP) 5COSC001W

Informatics Institute of Technology
Department of Computing
Module: 5COSC001W – Object Oriented Programming
Coursework 1
Date of submission: 18/11/2016
UOW ID: w1608504
IIT Stu. ID : 2015337
Name: Ashenika Perera

Contents

Functional Requirements

A functional Requirement is a function of a system or it`s component. A function is described as a set of inputs, the behavior and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to do. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

- **Add Vehicle**

  I used this as a functional requirement. Because as the definition, User can add a Vehicle to the System. And also he/she has to add some details about vehicle. Such as Vehicle ID , Brand, Color…etc.

- **Display Number of free slots**

  I used this as a functional Requirement. Because User can see the Display. And it also a function of the system.

- **Delete Vehicle**

  I used this as a functional Requirement. Because User/Manager can Delete a Vehicle from the system. It is a function of the system too.

- **Display type of Vehicle**

  I used this as a functional Requirement. Because User/Manager can Display the type of the Vehicle after deleting a vehicle.

- **Print a list of Vehicles**

  The reason that I used this as a functional Requirement, Because User can Print a list of current Vehicles if he/ she needed.

- **Print the Statistics**

  The reason that I used this as a functional Requirement, Because User can Print some Statistics as a list that shows Vehicle IDs and types. If he/ she needed.

- **Display Percentage of Vehicles**

  The User can see the percentage of each Vehicle types. That also a function of this system.

- **Display Last Parked Vehicle**

  The User can see the vehicle that was parked lastly. That is also a function of the system.

- **Display Charges**

  The User will be displayed the total price for the parking. After that User can Pay the money and take out the vehicle. The price is calculated for the spent hours.

Non-Functional Requirements

Functional requirements are supported by non-functional requirements which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

- **Chronologically Ordered List**

  As the definition, User can`t change the Ordered List, But only to view it. That`s why I used this as a Non-functional requirement.

- **Characters**

  The User can see the Characters of display of the system. But he/she can`t change it.

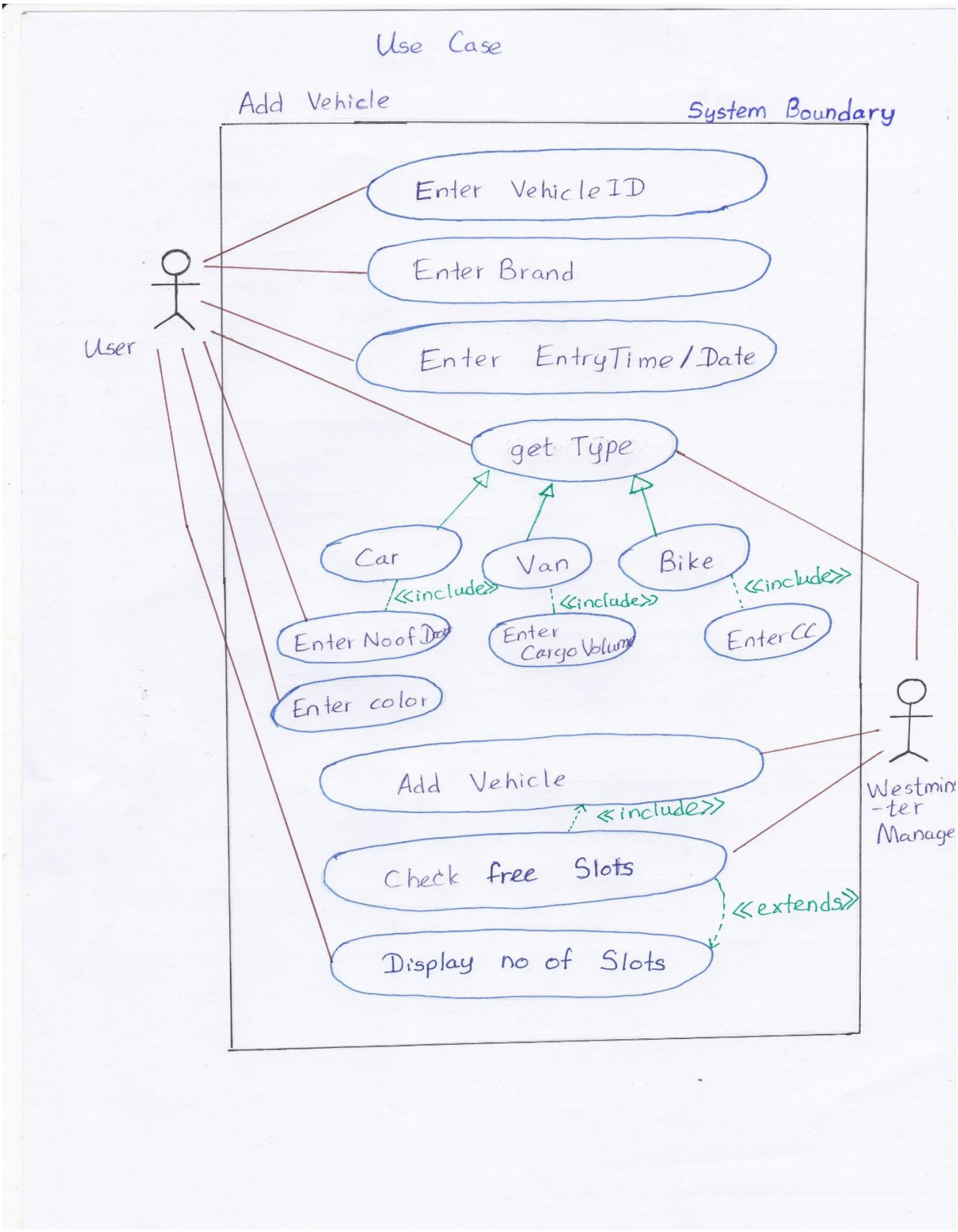
- **Sorted Lists**

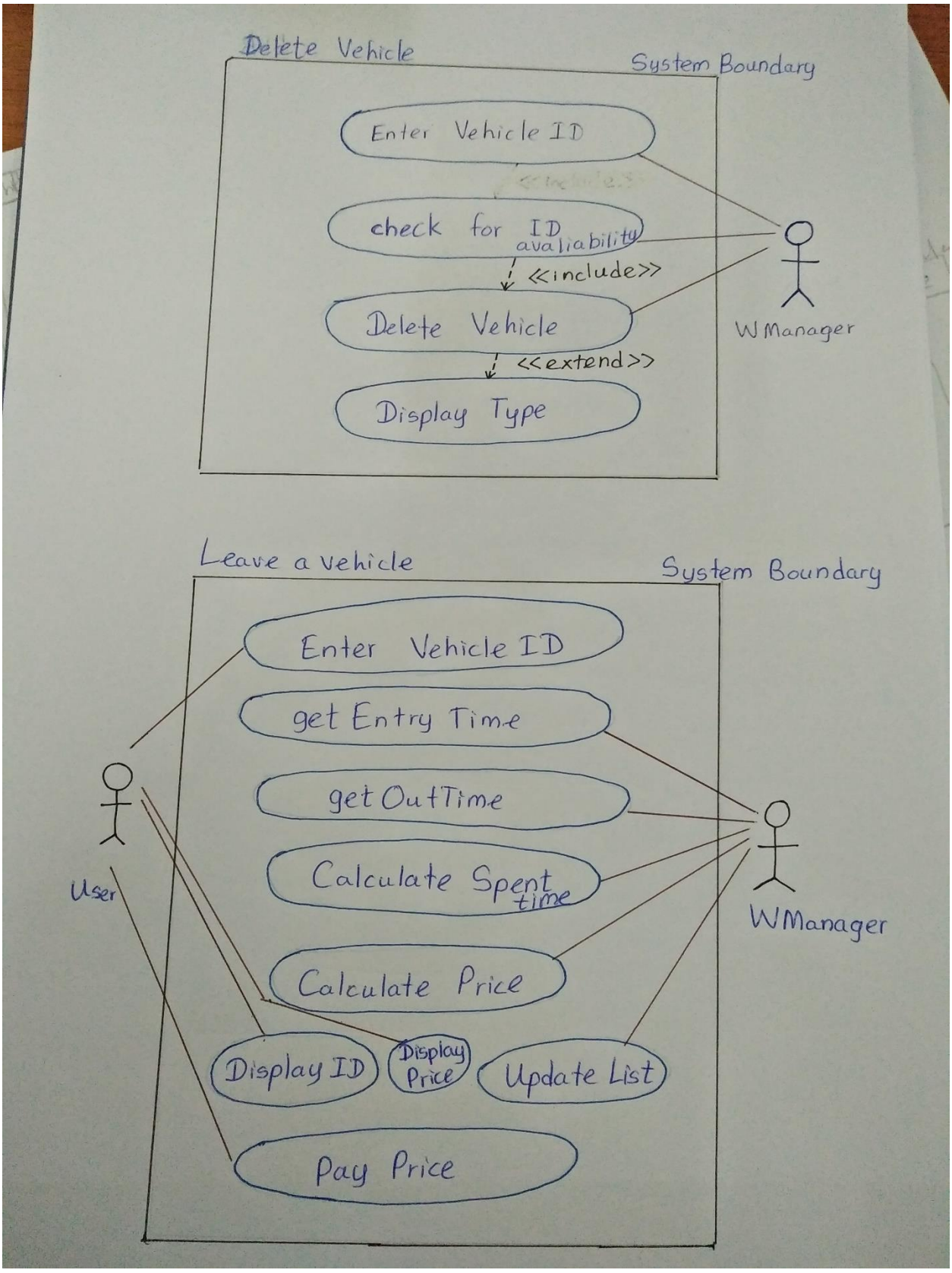
  The User can see the sorted list, But It is sorted by the system. Not by the user. So I used this as a Non-Functional Requirement.

- **Remaining number of Slots**

  The User can see the number of Slots. But It is Default for the System.

Use Case Diagrams




Delete Vehicle
System Boundary
Enter Vehicle ID
check for ID availability
<<include>>
Delete Vehicle
<<extend>>
Display Type
WManager
Leave a vehicle
System Boundary
Enter Vehicle ID
get Entry Time
get OutTime
Calculate Spent time
Calculate Price
Display ID
Display Price
Update List
Pay Price
User
WManager

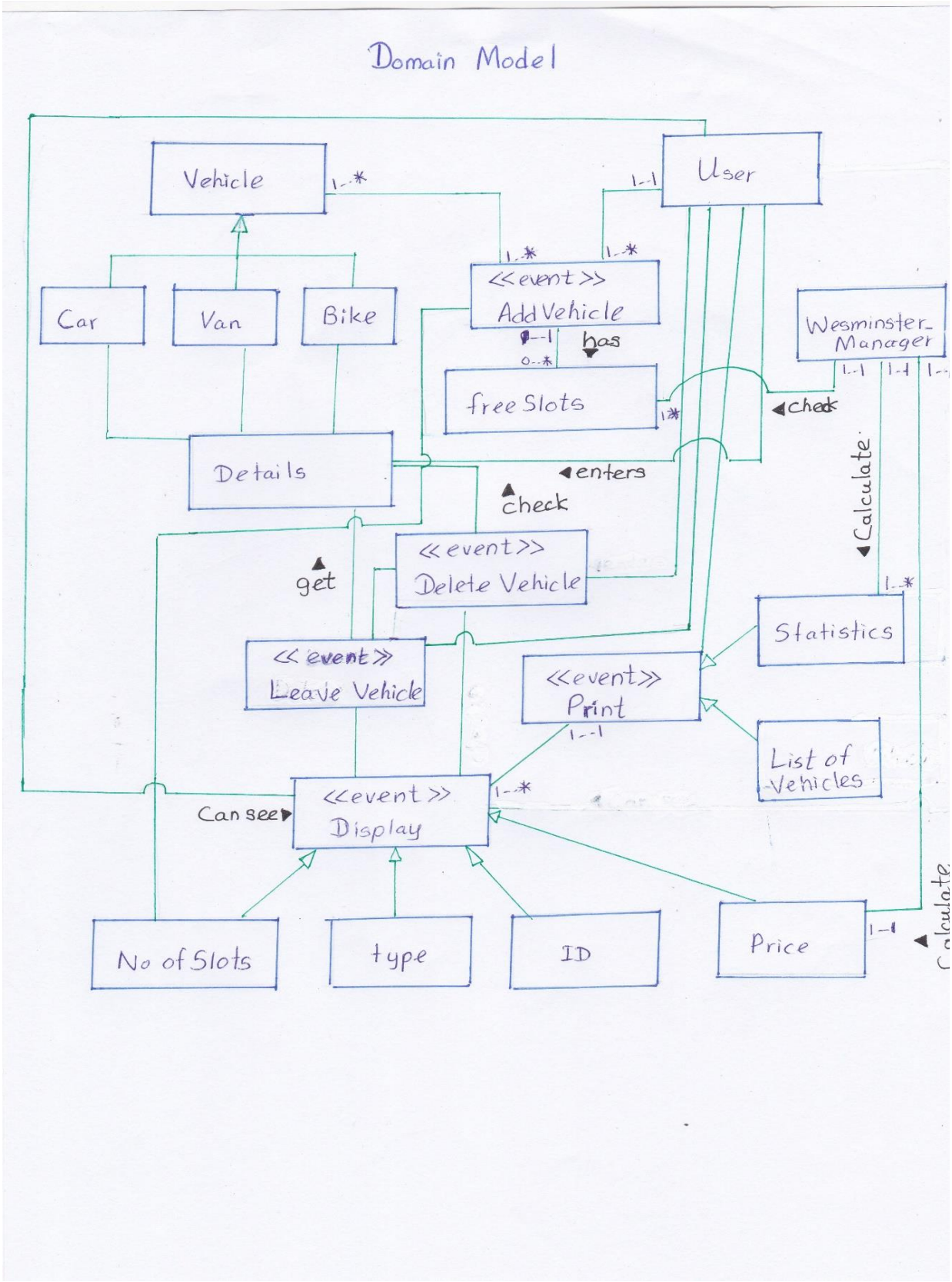
Use Case Description

Use Case Description for Add Vehicle		
Name	Add Vehicle	
Use Case Number	001	
Priority	high	
Participating Actors	User, Westminster Manager	
Pre-Conditions	Select “A” for Add vehicle	
Triggering Event	--	
Main Flow of Events	Actor : 2.enters ID 5. Enters Brand 7.Enters EntryTime & Date 10.Enters Details	System : 1.Prompts for ID 3.add ID to List 4.prompts for Brand 6.prompts for Entry Time/ Date 8.Get type 9.prompt for each details 11.Add Vehicle & details for the List 12.check for free slots 13.display no of slots
Alt Flow 1		11. 12.”Slots are full” 13. Go To 1.
Post Conditions	Select another Options	
Inclusions	admin	

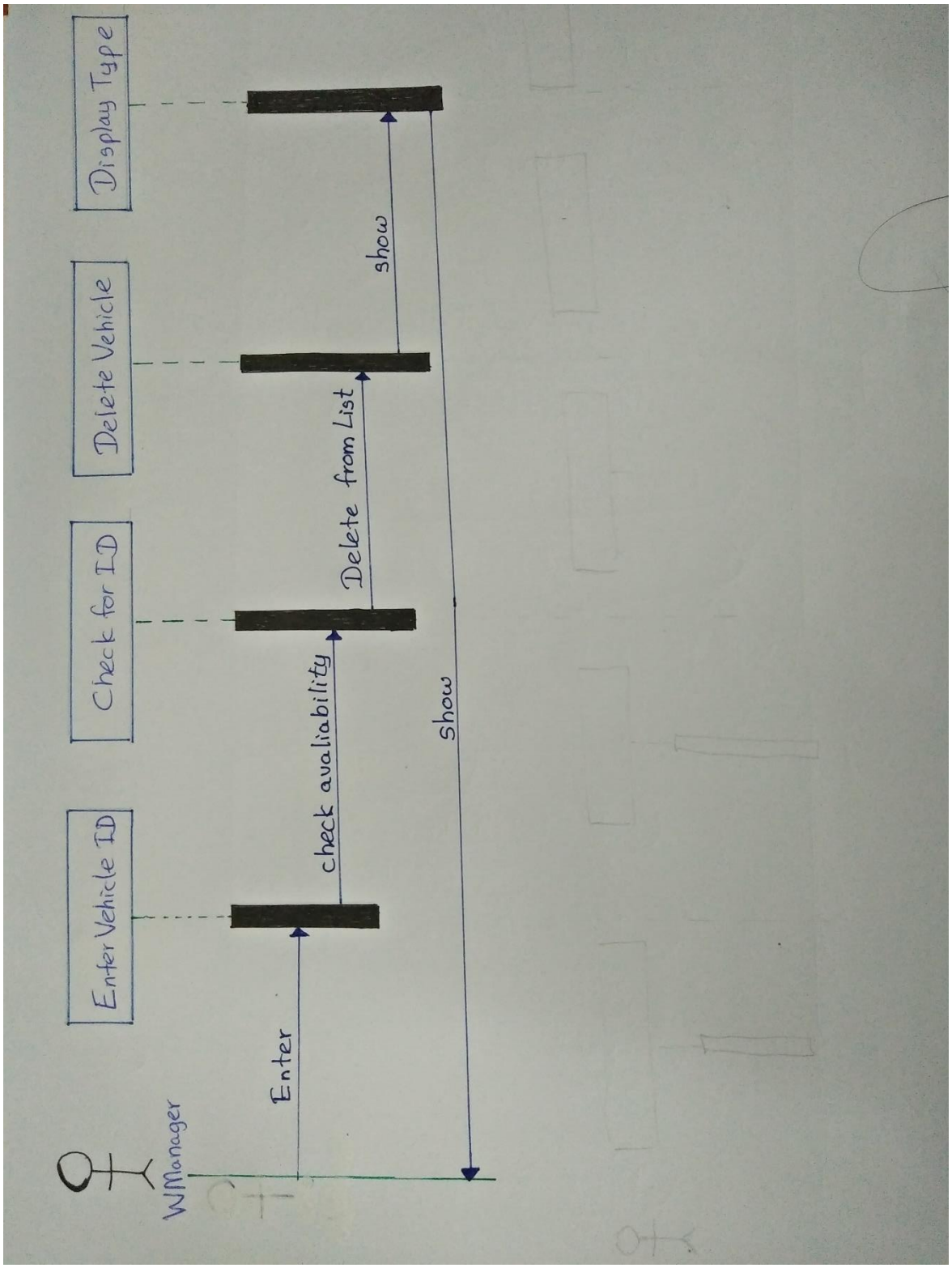
Use Case Description for Delete Vehicle		
Name	Delete Vehicle	
Use Case Number	002	
Priority	high	
Participating Actors	Westminster Manager	
Pre-Conditions	Select “D” for Add vehicle	
Triggering Event	--	
Main Flow of Events	Actor : 2.enters ID 5. Enters Brand	System : 1.Prompts for ID 3.Check for ID availability 4.Delete Vehicle from list 6.Check for the type 7.Display Type
Alt Flow 1		4. 6.”ID is not avaliable” 13. Go To 1.
Post Conditions	Select another Options	
Inclusions	admin	

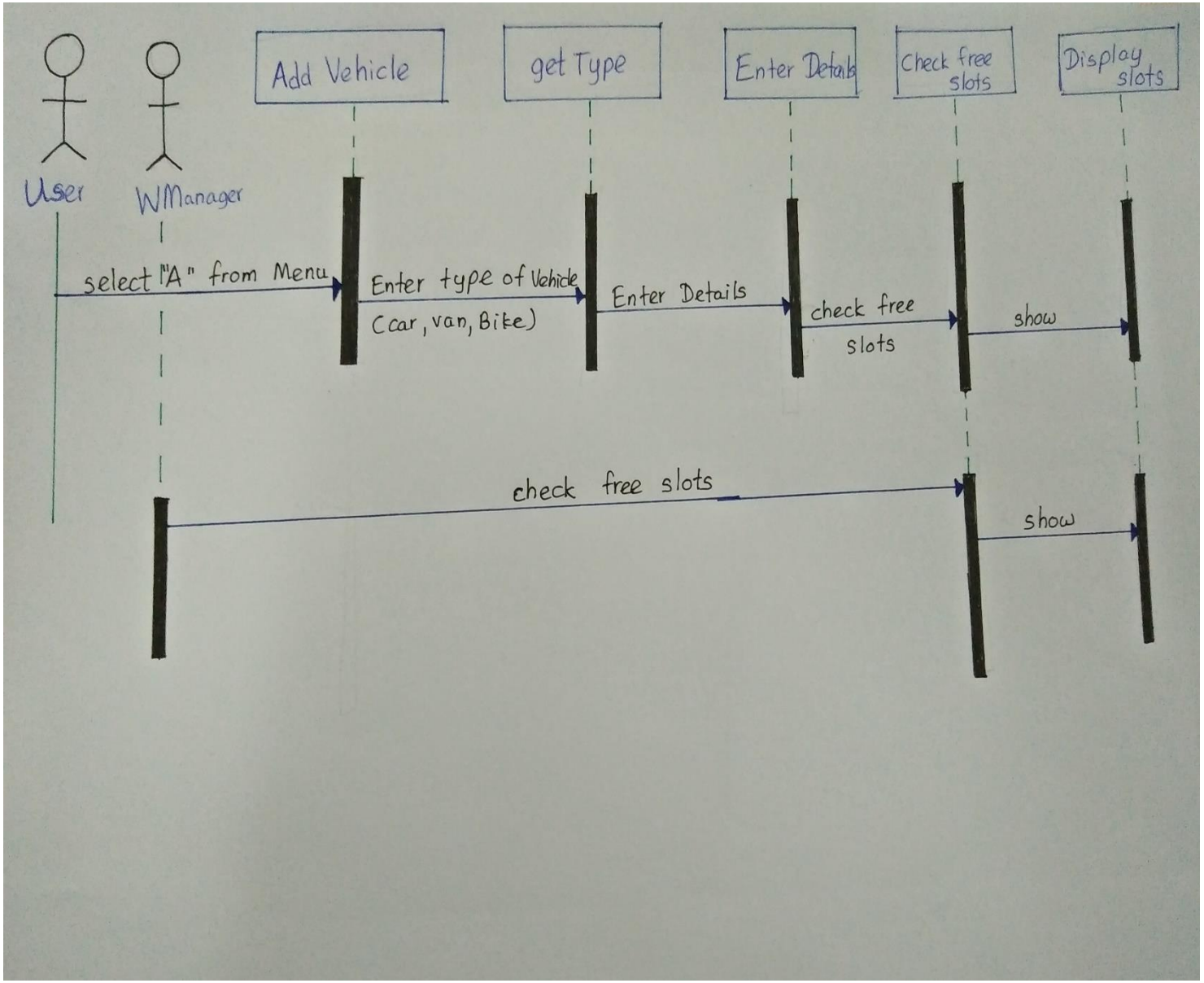
Use Case Description for Leave Vehicle		
Name	Leave Vehicle	
Use Case Number	003	
Priority	high	
Participating Actors	User, Westminster Manager	
Pre-Conditions	Select “L” for Add vehicle	
Triggering Event	--	
Main Flow of Events	Actor : 2.enters ID 5. Enters Out Time 10.Prompt system for Update List 13.Pay Price	System : 1.Prompts for ID 3.Get Entry Time 4.Prompt for Out Time 6.Calculate Spent Time 7.Calculate Price 9.Display ID 11.Display Price 12.Update List
Alt Flow 1	13.Credit card or Cash	
Post Conditions	Select another Options	
Inclusions	admin	

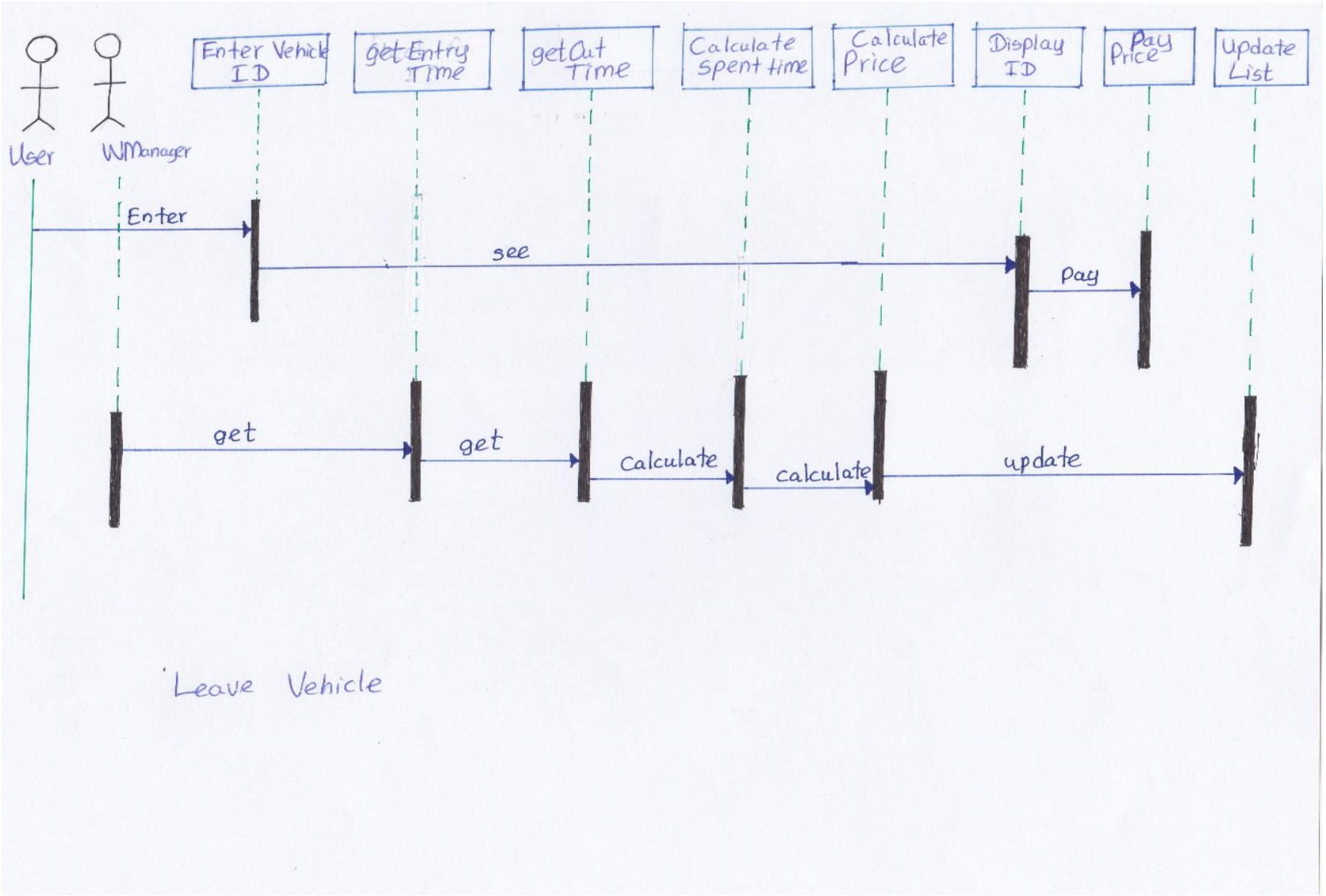
Domain Model



Sequence Diagrams

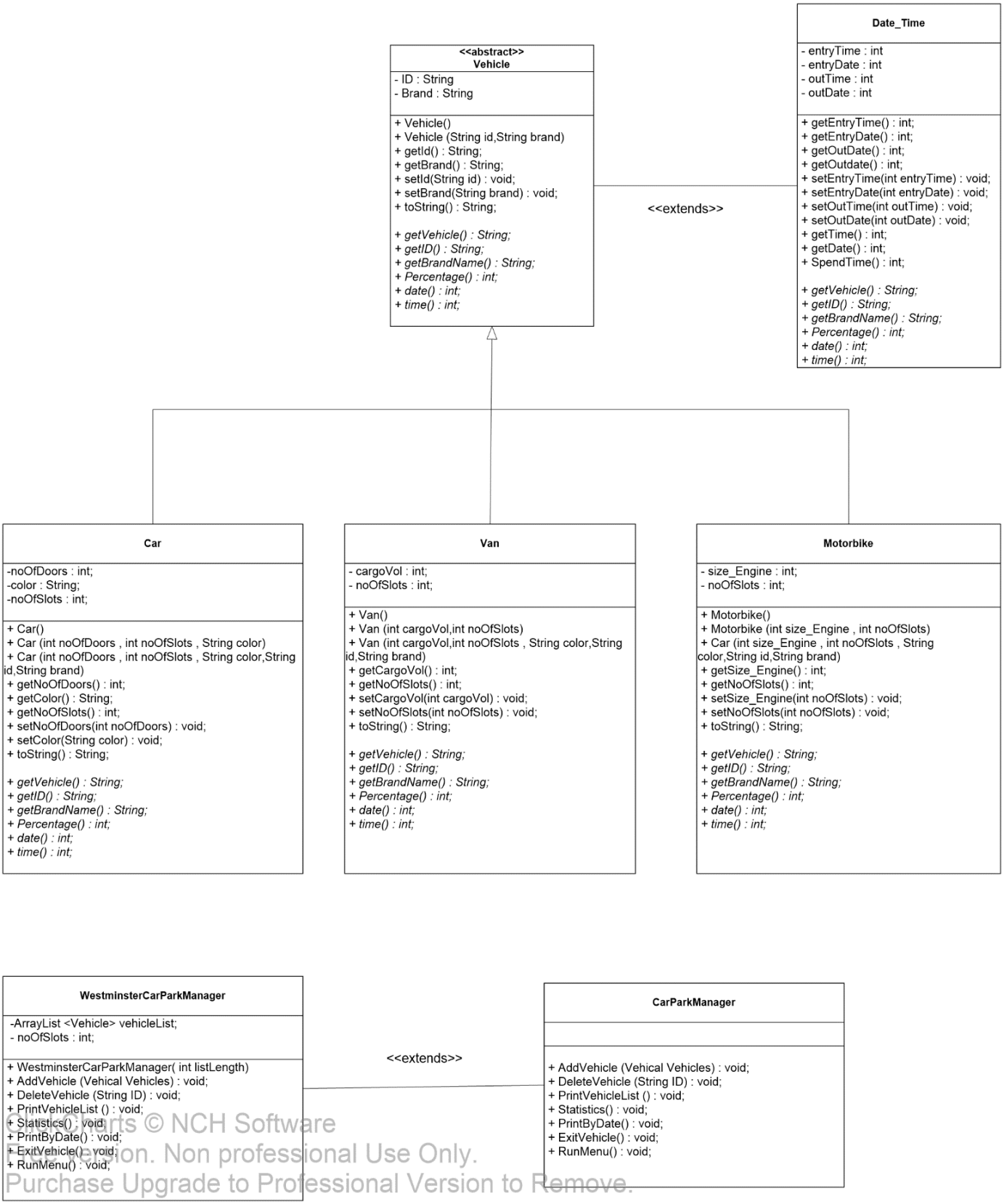



User
WManager
Add Vehicle
get Type
Enter Details
Check free slots
Display slots
select "A" from Menu
Enter type of Vehicle
(Car, van, Bike)
Enter Details
check free
slots
show
check free slots
show



'Leave Vehicle

Class Diagram



Screen Shots

```
Please Select Below Options
A : to Add a new vehicle
D : to delete a vehicle
P : Print the list of current vehicles
S : Print Some Statistics
B : Print By Date
L : Leave a Vehicle
E : Exit Menu
```

```
A
Please Select one Vehicle type from Below
C : for a Car
V : for a Van
M : for a Motorbike
C
Enter Vehicle Number
KR5623
Enter the Brand
Toyota
Enter entry Time in 0000 (24 hours) format
1230
Enter entry Date in month day format
0525
Enter number of doors in the car
5
Enter color of the car
Black
```

```
D
Enter Vehicle Number
KR5623
Deleted Vehicle is a Car
```

```
P
Vehicle = Car, Brand = Toyota, ID = KR5623
Vehicle = Van, Brand = Mazda, ID = HI4512
```

Testing

Black Box Testing

No	Input	Expected Output	Actual Output	Bug
1	Select a Letter from the menu (A,D,P,S,P,L,E) “A”	Please Select one Vehicle type from Below	Please Select one Vehicle type from Below	No
2	Select a Letter from the menu (A,V,E,D,F,S,L,O,X) “a”	Please Select one Vehicle type from Below	Please Select one Vehicle type from Below (because ignoreCase)	No
	Add Vehicle Please Select one Vehicle type from Below			
4	“C” for Car	Enter Vehicle Number :	Enter Vehicle Number : KR5623	No
5		Enter the Brand :	Enter the Brand : Toyota	No
6		Enter entry Time in 0000 (24 hours) format:	Enter entry Time in 0000 (24 hours) format : (1230)	No
7		Enter entry Date in month day format :	Enter entry Date in month day format : (0512)	No
		Enter number of doors in the car:	Enter number of doors in the car : (0512)	No
		Enter color of the car :	Enter color of the car: (Black)	No
	Print the List of Current Vehicles			
10	“P”	Vehicle = Car, Brand = Toyota, ID = KR5623	Vehicle = Car, Brand = Toyota, ID = KR5623	No
	Delete Vehicle			
8	Enter Vehicle Number (KR5623)	Deleted Vehicle is a Car	Deleted Vehicle is a Car	No

	Print Statistics			
	“S”	Percentage of Cars = 5% Vehicle that Parked for Long time is Vehicle{id=KR5623brand=toyota} Last Vehicle Parked is = KR5623	Percentage of Cars = 5% Vehicle that Parked for Long time is Vehicle{id=KR5623brand=toyota} Last Vehicle Parked is = KR5623	No

White Box Testing

Serial No	Function	Condition	Path
01	Add Vehicle	If (Add Vehicle is clicked)	Path 1 (True) Validate New Customer Path 2(False) Stop
02	Delete Vehicle	If(Delete Vehicle is clicked after entering the vehicle ID)	Path 1 (True) Validate Vehicle name Path 2(False) Stop
03	Print a List of Current Vehicles	If (Print List is selected)	Path 1 (True) Display: List Path 2 (False) Stop

Additional Task

```
interface i {
   public abstract void m1();
   public abstract void m2();
   default public void m3(String Meth){
      System.out.println("Default Method");
   }
}
interface i2 {
    void m1();
    void m2();
    default void m3(){
       System.out.println("Default Method!!");
    }
}
public class SirTask implements i , i2{

   public void m1(String Meth){
      System.out.println("m1 Method");
   }

   public void m2(){
      System.out.println("m2 Method");
   }

   public static void main(String[] args) {
      SirTask t = new SirTask();
      t.m1();
      t.m2();
      t.m3();
   }

   @Override
   public void m1() {
      System.out.println("Overrided m1 Method");
   }

}
```