

## Contents

Vehicle Fleet System(GemStone/Group-7 Version History) .....	2
Team Members and their Responsibilities .....	3
Problem statement & Requirement definition:.....	3
Vehicle fleet service:.....	3
Features implemented in REST API: .....	4
Entity Relationship Model .....	5
Object Model Diagram (optional).....	6
Normalization .....	7
Table Definitions and Data Contents.....	8
SQL Statements .....	11
Software Stack.....	14
User Interfaces and Database connectivity.....	15
Vehicles.....	16

## Vehicle Fleet System(GemStone/Group-7 Version History)

Version Number	Date	Author/Owner	Description of Change
0.1	30-Aug-2021	Sree Harshitha	Initial Setup and first Draft
0.2	31-Aug-2021	Niraj Chadrabhan	Problem Specification
0.3	05-Sept-2021	Sobha Nand Das	SQL Queries
0.4	14-Sept-2021	Yogesh Kumar	Rest API
0.5	28-Sept-2021	Rupesh Mittal	Table Defination
0.6	29-Sept-2021	BalKrishnan P	Normalization
0.7	25-Oct-2021	Khushnu Prashant	Model Diagram
0.8	27-Oct-2021	Ravi R	Normalization
0.9	06-Nov-2021	Ashrith Shetty	Rest API screenshot
1.0	07-Nov-2021	Surya Rayudu	Table Defination
1.1	08-Nov-2021	Anurag Kulshrestha	Overall review and ER Digram

## Team Members and their Responsibilities

Project Phase	Contribution By	Responsibility
Ideation	All team members	Came up with project idea and finalized Vehicle Fleet System as unique problem which no one solved
Functional Requirement	Anurag, Harishitha, Neeraj, Sobhanand, Khushbu, Surya	Problem definition and functional definitions
Entity Identification	All team members	To find out all required entities for problem domain
ER Diagram	Anurag	Write ER Diagram
Model Diagram	Khushbu, Sobhanand	Write up model diagram for the Vehicle Fleet System
Normalization	BalKrishnan P, Ravi R	Check all entity tables and their state corresponding to normalization
Service Code	All Team members	Write up code for all entity classes their invocation

## Problem statement & Requirement definition:

### Vehicle fleet service:

Group 7 team came up with service-based product called "Vehicle Fleet service" to help customer when they want transport vehicle similar to OLA or UBER apps. Vehicle Fleet service hereafter used as "VFS" in this document. VFS allow customer to book transport vehicle based on their requirements. Vehicle will be provided by fleet vendors. Our service will match the customer requirement and available vehicle. There are three types of user in this app.

Firstly, Customer who will avail the service and book transport vehicle from one place to another. He can view the vehicle for a particular route. He can view its rating and cost. He can track the trip.

Secondly, Fleet vendor who will register their vehicle and driver information in the portal. They can approve/reject the booking. He can update the vehicle problem and stop its visibility in the system. He can end the trip.

Thirdly, Administration who will maintain the production system. He will help the fleet vendor in onboarding. He can view any booking and update cost for each vehicle type. He can generate report of all the trips.

Consider the following example explaining how the VFS service ease the life of customer and fleet vendor:

Suppose you want to shift your home, you have requirement of truck for 2 BHK home.

- You can search the available vehicle between source and destination and cost associated with them.
- You can select vehicle, source and destination and raise the booking request for a particular date.
- Fleet vendor will get this request, he can approve this request or reject it.
- After approval, invoice will be generated, and customer can do payment.
- During journey the location will updated, so that customer can track his booking.
- Customer can do payment after delivery also and write review on the vendor and booking.

Questions:

1. Who will be the users?

- Customer
  - Fleet vendor
  - Administrator
2. What are the benefits of this application?
    - Easy for customer to book good quality transport vehicle with good vendor support.
    - Easy for fleet vendor in managing the vehicle and booking orders.
    - Easy for tracking booking and payment for both customer and vendor.
  3. List of functions and features of the application?
    - Searching and Booking vehicle
    - Tracking the trip
    - Managing the fleet of vehicle
  4. How many users will use it simultaneously?
    - 2 types of user can use the application.
    - 100 users can access this application simultaneously at this point which can be further scaled based on infrastructure.

## Features implemented in REST API:

### Customer:

- Search fleets available from location (source) to location (destination)
- Results Will be all vehicle along with vehicle type and cost per km and carry capacity in kg and review information
- Book the vehicle (now and scheduled)
- Viewing Current trip Status (location of current state vehicle, live location)
- Payment Options – Cash
- Review Comments

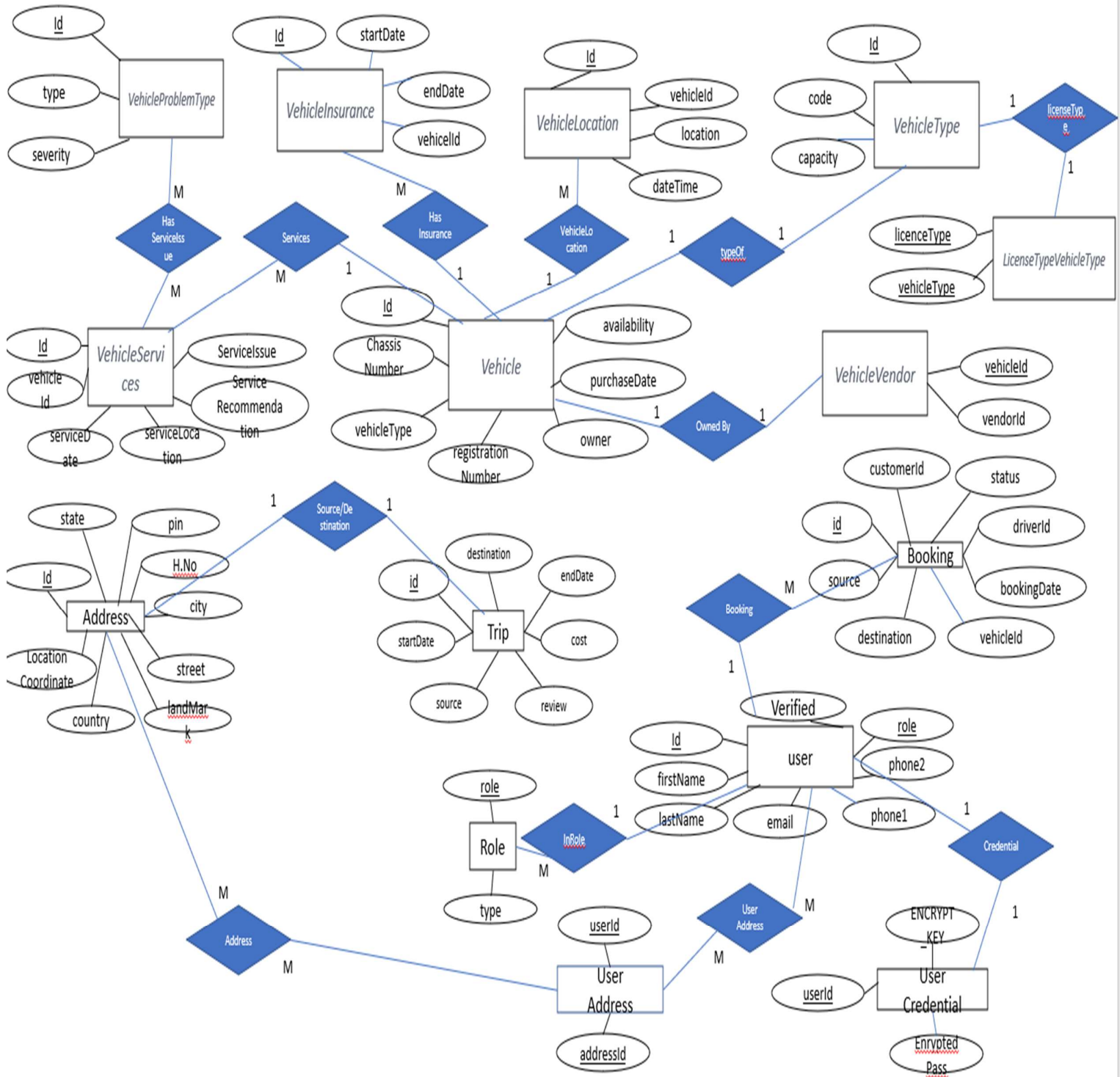
### Fleet Vendor:

- Add, Delete Vehicle
- Able to Add Vehicle Service Records
- Last Service issues for Vehicle
- Add/Delete Driver Information
- View booking and approving it.
- Viewing On going Trip Information
- Customer Review
- Update of vehicle tracking information
- Reports about Vehicle health like problem occurred, date and type
- End Trip

### Administrator:

- Adding per km cost for vehicle type
- Adding BHK vs suggested vehicle
- Viewing Current ongoing trips
- Monthly Summary of all trips
- Weekly Summary of all trips
- Reports on various trips
- Vehicle type addition
- Add/Delete Type of Vehicle Problem

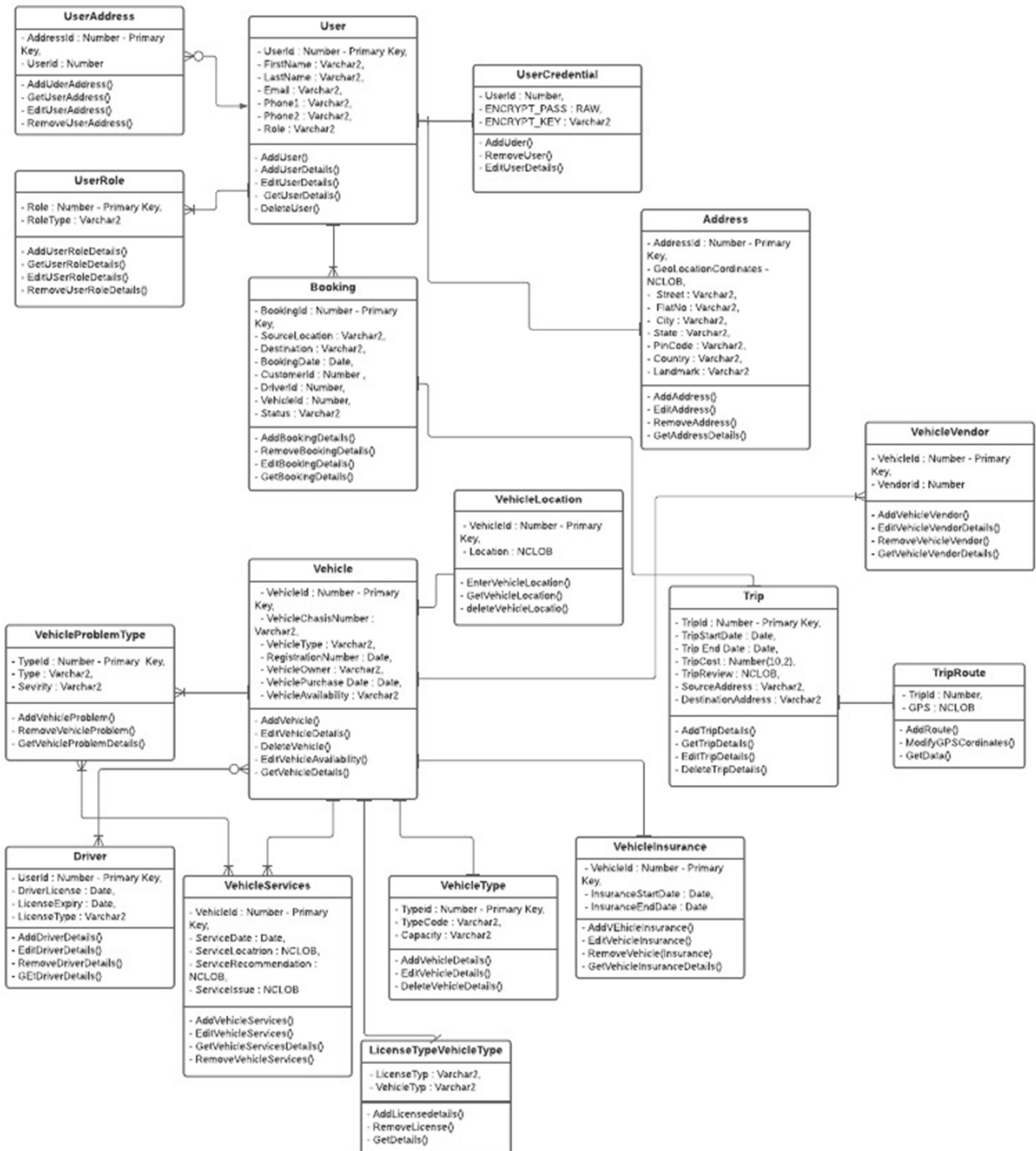
## Entity Relationship Model



<https://github.com/Group7-BITS/Vehicle-Fleet-System/blob/main/documents/ERDiagram.pptx>

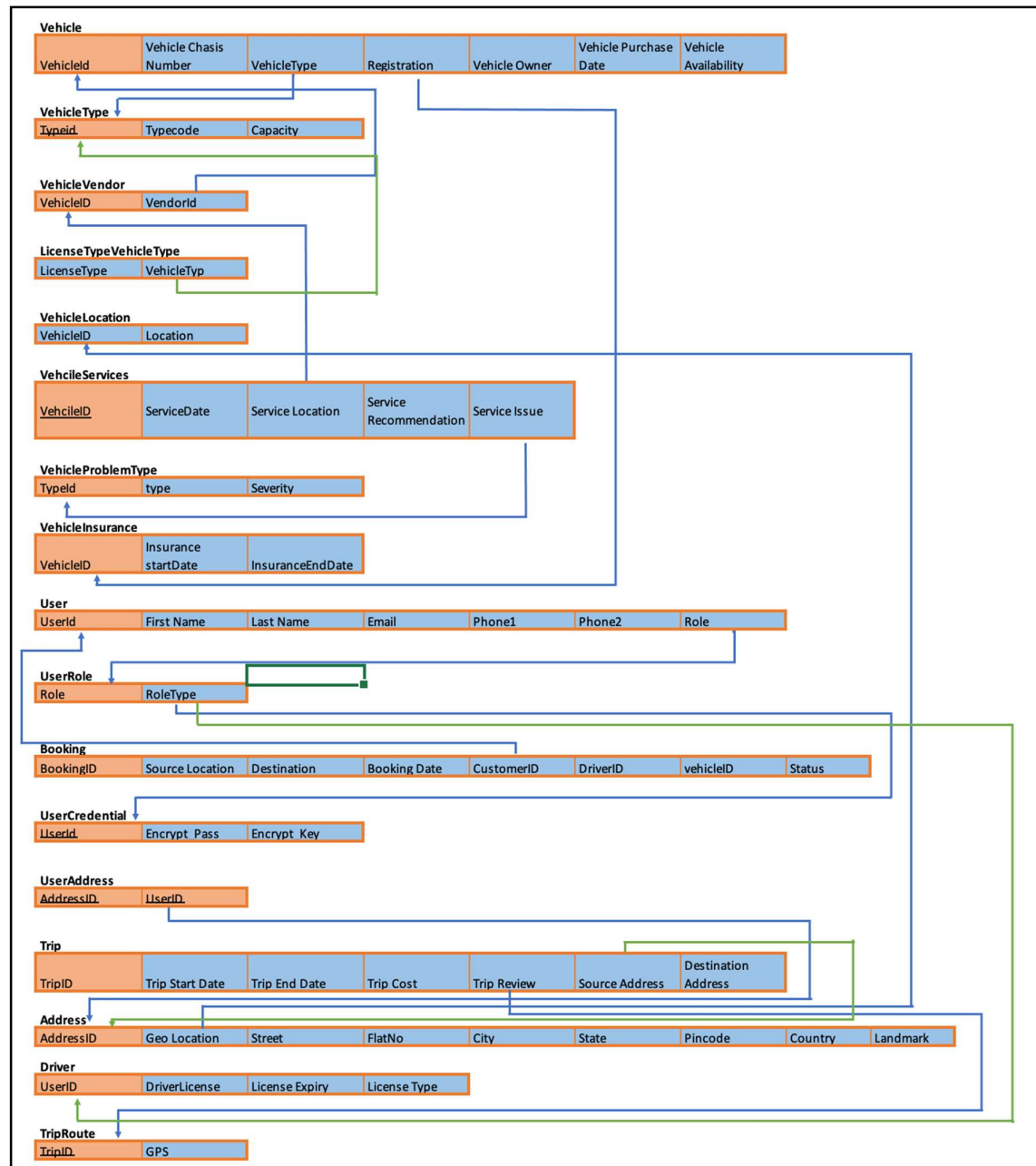
## Object Model Diagram (optional)

<https://github.com/Group7-BITS/Vehicle-Fleet-System/blob/main/documents/ModelDigram.jpeg>



## Normalization

Include the below for all the tables.



- We analysed all the 17 relations that were formed as a part of relational design .Here is what we observed:
  - All the 17 relations are already in 1NF as there were no attributes that are multivalued and stored in a single table .Hence we concluded our relational schema is in 1NF.

- We also verified and found that all the tables are also in 2NF . That is because there are no relations where any non key attribute is partially dependent on the key attribute.
  1. All of our relations have only one attribute as primary key . Since there is no composite key in these relations the concept of partial dependency does not arise here .
  2. However , table user had partial dependency, so the table was decomposed into User table, Address table and UserAddress table Hence we verified that none of the non key attributes are partially dependent on the composite key (AddressID,UserID).
- Next , we navigate to check if all our relations are in 3 NF or not. 3NF is based on concept of transitive dependency . We have successfully proved that all our relations are now in 3NF.
- We move one step ahead to check for BCNF . As per our observation all our relations are already in BCNF as we have no non-prime attribute that determines another non prime attribute in any of our relations . In all the relations concepts of fully functional dependency of prime attribute holds . Hence our relations are in BCNF. Hence in conclusion , our schema is in BCNF

## Table Definitions and Data Contents

User

Name	Data type	Primary Key
User id	Number	Key
First name	varchar2(20)	
Last name	varchar2(20)	
Email	varchar2(50)	
Phone 1	varchar2(15)	
Phone 2	varchar2(15)	
Role	varchar2(20)	

VehicleType

Name	Data type	Primary Key
Type id	Number	Key
TypeCode	varchar2(20)	



Capacity	varchar2(3)	
----------	-------------	--

TripRoute

Name	Data type	Primary Key
TripId	Number	
GPS	NCLOB	

VehicleLocation

Name	Data type	Primary Key
VehicleId	Number	Key
Location	NCLOB	

LicenseTypeVehicleType

Name	Data type	Primary Key
LicenseTyp	Varchar2(100)	
VehicleTyp	Varchar2(100)	

Vehicle

Name	Data type	Primary Key
VehicleId	Number	Key
VehicleChasisNumber	Varchar2(20)	
VehicleType	Varchar2(20)	
RegistrationNumber	Date	
VehicleOwner	Varchar2(20)	
VehiclePurchase Date	Date	
VehicleAvailability	Varchar2(20)	

VehicleProblemType

Name	Data type	Primary Key
Typeld	Number	key
Type	Varchar2(20)	
Severity	Varchar2(20)	

VehicleInsurance

Name	Data type	Primary Key
VehicleId	Number	key
InsuranceStartDate	Date	
InsuranceEndDate	Date	

Driver

Name	Data type	Primary Key
UserId	Number	key
DriverLicense	Date	
LicenseExpiry	Date	
LicenseType	Varchar2(20)	

### Trip

Name	Data type	Primary Key
TripId	Number	Key
TripStartDate	Date	
Trip End Date	Date	
TripCost	Number(10,2)	
TripReview	NCLOB	
SourceAddress	Varchar2(200)	
DestinationAddress	Varchar2(200)	

### VehicleServices

Name	Data type	Primary Key
VehicleId	Number	Key
ServiceDate	Date	
ServiceLocatrion	NCLOB	
ServiceRecommendation	NCLOB	
ServiceIssue	NCLOB	

### UserAddress

Name	Data type	Primary Key
AddressId	Number	Key
UserId	Number	

### Address

Name	Data type	Primary Key
AddressId	Number	Key
GeoLocationCordinates	NCLOB	
Street	Varchar2(20)	
FlatNo	Varchar2(20)	
City	Varchar2(20)	
State	Varchar2(50)	
PinCode	Varchar2(20)	
Country	Varchar2(50)	
Landmark	Varchar2(200)	

### UserCredential

Name	Data type	Primary Key
UserId	Number	
ENCRYPT_PASS	RAW(200)	
ENCRYPT_KEY	Varchar2(100)	

### VehicleVendor

Name	Data type	Primary Key
VehicleId	Number	Key
VendorId	Number	

#### Booking

Name	Data type	Primary Key
BookingId	Number	key
SourceLocation	Varchar2(200)	
Destination	Varchar2(200)	
BookingDate	Date	
CustomerId	Number	
DriverId	Number	
VehicleId	Number	
Status	Varchar2(20)	

#### UserRole

Name	Data type	Primary Key
Role	Number	Key
RoleType	Varchar2(20)	

## SQL Statements

```
CREATE TABLE "User" (  
    "UserId" Number Primary Key,  
    "FirstName" Varchar2(20),  
    "LastName" Varchar2(20),  
    "Email" Varchar2(50),  
    "Phone1" Varchar2(15),  
    "Phone2" Varchar2(15),  
    "Role" Varchar2(20)  
);  
  
CREATE TABLE "TripRoute" (  
    "TripId" Number,  
    "GPS" NCLOB  
);  
  
CREATE TABLE "VehicleType" (  
    "Typeid" Number Primary Key,  
    "TypeCode" Varchar2(20),  
    "Capacity" Varchar2(3)  
);  
  
CREATE TABLE "VehicleLocation" (  
    "VehicleId" Number Primary Key,  
    "Location" NCLOB  
);  
  
CREATE TABLE "LicenseTypeVehicleType" (  
    "LicenseTyp" Varchar2(100),  
    "VehicleTyp" Varchar2(100)
```

```

);

CREATE TABLE "Vehicle" (
  "VehicleId" Number Primary Key,
  "VehicleChasisNumber" Varchar2(20),
  "VehicleType" Varchar2(20),
  "RegistrationNumber" Date,
  "VehicleOwner" Varchar2(50),
  "VehiclePurchase Date" Date,
  "VehicleAvailability" Varchar2(5)
);

CREATE TABLE "VehicleProblemType" (
  "TypeId" Number Primary Key,
  "Type" Varchar2 (20),
  "Sevirity" Varchar2(20)
);

CREATE TABLE "VehicleInsurance" (
  "VehicleId" Number Primary Key,
  "InsuranceStartDate" Date,
  "InsuranceEndDate" Date
);

CREATE TABLE "Driver" (
  "UserId" Number Primary Key,
  "DriverLicense" Date,
  "LicenseExpiry" Date,
  "LicenseType" Varchar2(20)
);

CREATE TABLE "Trip " (
  "TripId" Number Primary Key,
  "TripStartDate" Date,
  "Trip End Date" Date,
  "TripCost" Number(10,2),
  "TripReview" NCLOB,
  "SourceAddress" Varchar2(200),
  "DestinationAddress" Varchar2(200)
);

CREATE TABLE "VehicleServices" (
  "VehicleId" Number Primary Key,
  "ServiceDate" Date,
  "ServiceLocatrion" NCLOB,
  "ServiceRecommendation" NCLOB,
  "ServiceIssue" NCLOB
);

```

```

CREATE TABLE "UserAddress" (
  "AddressId" Number Primary Key,
  "UserId" Number,
  CONSTRAINT "FK_UserAddress.UserId"
    FOREIGN KEY ("UserId") REFERENCES "User"("UserId")
);

CREATE TABLE "Address" (
  "AddressId" Number Primary Key,
  "GeoLocationCoordinates" NCLOB,
  "Street" Varchar2(20),
  "FlatNo" Varchar2(20),
  "City" Varchar2(20),
  "State" Varchar2(50),
  "PinCode" Varchar2(20),
  "Country" Varchar2(50),
  "Landmark" Varchar2(200),
  CONSTRAINT "FK_Address.AddressId"
    FOREIGN KEY ("AddressId")
      REFERENCES "UserAddress"("AddressId")
);

CREATE TABLE "UserCredential" (
  "UserId" Number,
  "ENCRYPT_PASS " RAW (200),
  "ENCRYPT_KEY" Varchar2(100),
  CONSTRAINT USER_ID_REF FOREIGN KEY ("UserId") REFERENCES "User"("UserId")
);

CREATE TABLE "VehicleVendor" (
  "VehicleId" Number Primary Key,
  "VendorId" Number
);

CREATE TABLE "Booking" (
  "BookingId" Number Primary Key,
  "SourceLocation" Varchar2(200),
  "Destination" Varchar2(200),
  "BookingDate" Date,
  "CustomerId" Number ,
  "DriverId" Number,
  "VehicleId" Number,
  "Status" Varchar2(20),
  CONSTRAINT CustomerId_ref FOREIGN KEY ("CustomerId") REFERENCES
"User"("UserId"),
  CONSTRAINT DriverId_ref FOREIGN KEY ("DriverId") REFERENCES
"Driver"("UserId"),

```

```
CONSTRAINT VehicleId_ref FOREIGN KEY ("VehicleId") REFERENCES
"Vehicle"("VehicleId")
);

CREATE TABLE "UserRole" (
  "Role" Number Primary Key,
  "RoleType" Varchar2(20)
);
```

## Software Stack

- **Java (JDK 8)**
- **Spring Boot : (Can Handle 100 concurrent requests simultanously.)**
- **Postman**
- **GIT**
- **Maven**
- **Eclipse**
- **HSQL DB** HSQL DB open source free light weight database system which can be easily integrated with any technology and easy to scale and back up the data as well.
- **JPA Entity**
- **VFS** currently supporting 100 simultaneous users concurrently, 100 size database connection pools which can be further scaled if we deploy to cloud using multi region cloud deployment.

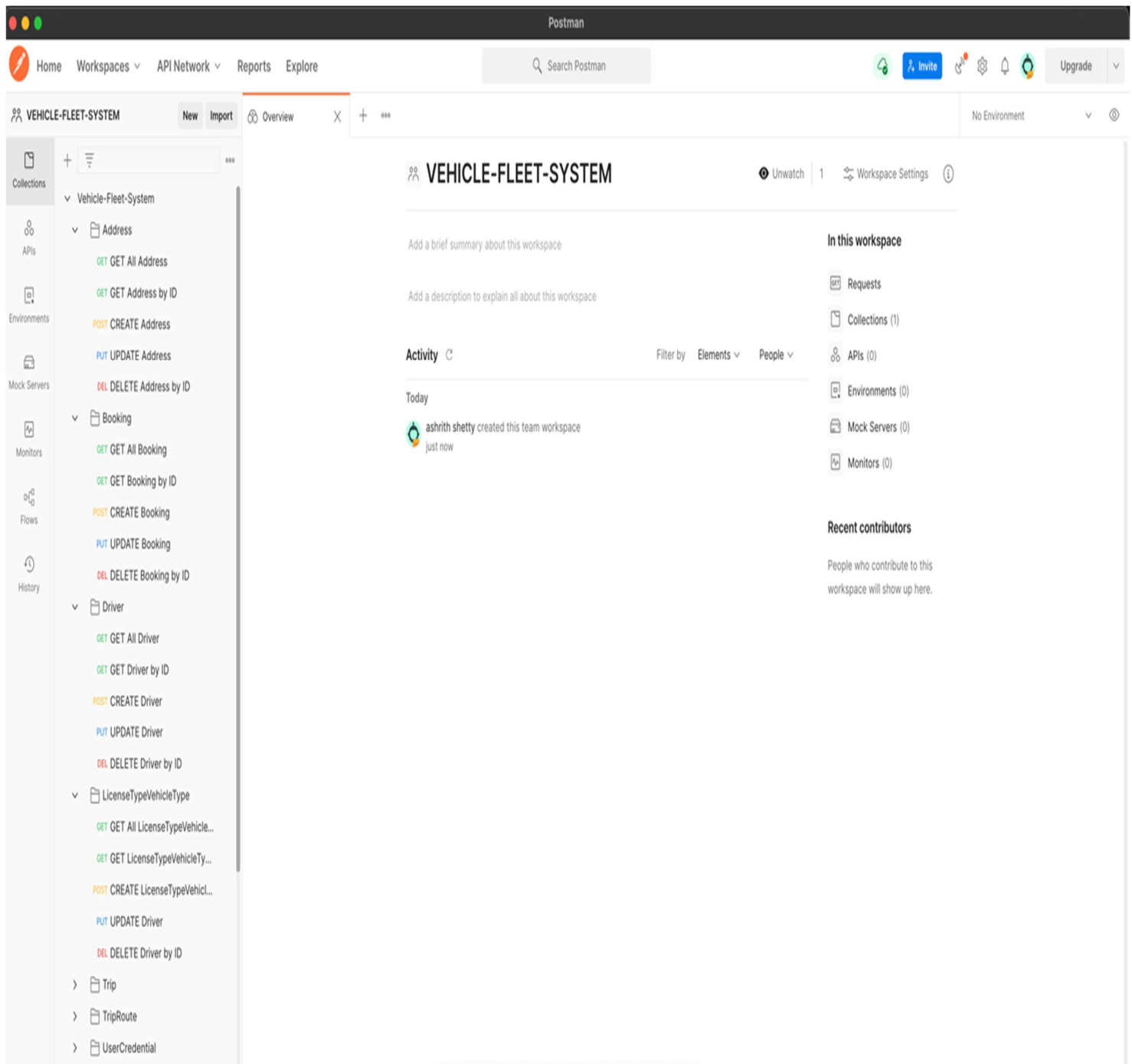
## User Interfaces and Database connectivity

# Vehicle Fleet System

This is written for all entity classes and below screen shot is for vehicle only.

For detail import below file into postman to see all entities all CRUD operations working

[https://github.com/Group7-BITS/Vehicle-Fleet-System/blob/main/documents/Vehicle-Fleet-System.postman\\_collection.json](https://github.com/Group7-BITS/Vehicle-Fleet-System/blob/main/documents/Vehicle-Fleet-System.postman_collection.json)



# Vehicles

## 1. Get All Vehicles

VEHICLE-FLEET-SYSTEM

NewImport

Collections

Vehicle-Fleet-System

Address

Booking

Driver

LicenseTypeVehicleType

Trip

TripRoute

UserCredential

Vehicle

GET GET All Vehicles

GET GET Vehicle by ID

POST CREATE Vehicle

PUT UPDATE Vehicle

DEL DELETE Vehicle by ID

User

User Role

VehicleInsurance

VehicleLocation

VehicleProblemType

VehicleServices

VehicleType

VehicleVendor

APIs

Environments

Mock Servers

Monitors

Flows

History

GET GET All Vehicles

Vehicle-Fleet-System / Vehicle / GET All Vehicles

GET

http://localhost:8080/vehicles

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE
-----	-------

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

{

" id": 1,

"chasisNumber": "CH-0567899",

"registrationNumber": "RG-UP074567",

"registrationStartDate": "2021-11-09T17:32:57.456+00:00"

},

{

" id": 2,

"chasisNumber": "CH-879763536",

"registrationNumber": "RG-TG089765",

"registrationStartDate": "2021-11-09T17:32:57.498+00:00"

}

}



## 2. Get Vehicle By ID

Booking

Vehicle-Fleet-System

New

Import

Collections

APIs

Environments

Mock Servers

Monitors

Flows

History

+

Vehicle-Fleet-System

> Address

> Booking

> Driver

> LicenseTypeVehicleType

> Trip

> TripRoute

> UserCredential

> Vehicle

GET GET All Vehicles

GET GET Vehicle by ID

POST CREATE Vehicle

PUT UPDATE Vehicle

DEL DELETE Vehicle by ID

> User

> User Role

> VehicleInsurance

> VehicleLocation

> VehicleProblemType

> VehicleServices

> VehicleType

> VehicleVendor

GET GET All Vehicles

GET GET Vehicle by ID

+ ...

Vehicle-Fleet-System / Vehicle / GET Vehicle by ID

GET

http://localhost:8080/vehicles/1

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsS

Query Params

KEY	VALUE
Key	Value

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 1,
3   "chasisNumber": "CH-0567899",
4   "registrationNumber": "RG-UP074567",
5   "registrationStartDate": "2021-11-09T17:32:57.456+00:00"
6 }
```

### 3. Create Vehicle

VEHICLE-FLEET-SYSTEM

NewImport

Collections

APIs

Environments

Mock Servers

Monitors

Flows

History

+

☰

Vehicle-Fleet-System

> Address

> Booking

> Driver

> LicenseTypeVehicleType

> Trip

> TripRoute

> UserCredential

> Vehicle

> User

> User Role

> VehicleInsurance

> VehicleLocation

> VehicleProblemType

> VehicleServices

> VehicleType

> VehicleVendor

GET GET All Vehicles

GET GET Vehicle by ID

POST CREATE Vehicle

PUT UPDATE Vehicle

DEL DELETE Vehicle by ID

Vehicle-Fleet-System / Vehicle / CREATE Vehicle

POST

http://localhost:8080/vehicles

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1 {

2     "chassisNumber": "CH-0567800",

3     "registrationNumber": "RG-UP0887764",

4     "registrationStartDate": "2021-11-10T17:32:57.456+00:00"

5 }

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

VEHICLE-FLEET-SYSTEM

NewImport

Collections

Vehicle-Fleet-System

- Address
- Booking
- Driver
- LicenseTypeVehicleType
- Trip
- TripRoute
- UserCredential
- Vehicle
  - GET GET All Vehicles
  - GET GET Vehicle by ID
  - POST CREATE Vehicle
  - PUT UPDATE Vehicle
  - DEL DELETE Vehicle by ID
- User
- User Role
- VehicleInsurance
- VehicleLocation
- VehicleProblemType
- VehicleServices
- VehicleType
- VehicleVendor

GET GET All Vehicles

GET GET Vehicle by ID

POST CREATE Vehicle

+

...

Vehicle-Fleet-System / Vehicle / GET All Vehicles

GET

http://localhost:8080/vehicles

ParamsAuthorizationHeaders (9)Body●Pre-request ScriptTestsSettings

Query Params

KEY	VALUE
-----	-------

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   {
3     "id": 1,
4     "chassisNumber": "CH-0567899",
5     "registrationNumber": "RG-UP074567",
6     "registrationStartDate": "2021-11-09T17:32:57.456+00:00"
7   },
8   {
9     "id": 2,
10    "chassisNumber": "CH-879763536",
11    "registrationNumber": "RG-TG089765",
12    "registrationStartDate": "2021-11-09T17:32:57.498+00:00"
13  },
14  {
15    "id": 20,
16    "chassisNumber": "CH-0567800",
17    "registrationNumber": "RG-UP0887764",
18    "registrationStartDate": "2021-11-10T17:32:57.456+00:00"
19  }
20 }
```

#### 4. Update Vehicle

VEHICLE-FLEET-SYSTEM   New   Import   GET GET All Vehicles   GET GET Vehicle by ID   POST CREATE Vehicle   PUT UPDATE Vehicle

Vehicle-Fleet-System / Vehicle / UPDATE Vehicle

PUT   http://localhost:8080/vehicles/20

Params   Authorization   Headers (9)   Body   Pre-request Script   Tests   Settings

none   form-data   x-www-form-urlencoded   raw   binary   GraphQL   JSON

```
1 {
2   "chassisNumber": "CH-0567801",
3   "registrationNumber": "RG-UP0887765",
4   "registrationStartDate": "2021-11-10T17:32:57.456+00:00"
5 }
```

Body   Cookies   Headers (3)   Test Results

Pretty   Raw   Preview   Visualize   Text

```
1
```

VEHICLE-FLEET-SYSTEM   New   Import   GET GET All Vehicles   GET GET Vehicle by ID   POST CREATE Vehicle   PUT UPDATE Vehicle   +

Vehicle-Fleet-System / Vehicle / GET Vehicle by ID

GET   http://localhost:8080/vehicles/20

Params   Authorization   Headers (9)   Body   Pre-request Script   Tests   Settings

Query Params

KEY	VALUE
Key	Value

Body   Cookies   Headers (5)   Test Results

Pretty   Raw   Preview   Visualize   JSON

```
1 {
2   "id": 20,
3   "chassisNumber": "CH-0567801",
4   "registrationNumber": "RG-UP0887765",
5   "registrationStartDate": "2021-11-10T17:32:57.456+00:00"
6 }
```

## 5. Delete Address

VEHICLE-FLEET-SYSTEM

NewImport

GET GET All VehiclesGET GET Vehicle by IDPOST CREATE VehiclePUT UPDATE VehicleDEL DELETE Vehicle by...

Collections

+Vehicle-Fleet-System

>Address

>Booking

>Driver

>LicenseTypeVehicleType

>Trip

>TripRoute

>UserCredential

>Vehicle

GET GET All Vehicles

GET GET Vehicle by ID

POST CREATE Vehicle

PUT UPDATE Vehicle

DEL DELETE Vehicle by ID

>User

>User Role

>VehicleInsurance

>VehicleLocation

>VehicleProblemType

>VehicleServices

VEHICLE-FLEET-SYSTEM

NewImport

GET GET All VehiclesGET GET Vehicle by IDPOST CREATE VehiclePUT UPDATE VehicleDEL DELETE Vehicle by...

Collections

+Vehicle-Fleet-System

>Address

>Booking

>Driver

>LicenseTypeVehicleType

>Trip

>TripRoute

>UserCredential

>Vehicle

GET GET All Vehicles

GET GET Vehicle by ID

POST CREATE Vehicle

PUT UPDATE Vehicle

DEL DELETE Vehicle by ID

>User

>User Role

>VehicleInsurance

>VehicleLocation

>VehicleProblemType

>VehicleServices

>VehicleType

>VehicleVendor

Vehicle-Fleet-System / Vehicle / DELETE Vehicle by ID

DELETEhttp://localhost:8080/vehicles/20

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE
Key	Value

BodyCookiesHeaders (4)Test Results

PrettyRawPreviewVisualizeText

1

Vehicle-Fleet-System / Vehicle / GET All Vehicles

GEThttp://localhost:8080/vehicles

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE
-----	-------

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON

1{

2  {id: 1,

3  "chassisNumber": "CH-0567899",

4  "registrationNumber": "RG-UP074567",

5  "registrationStartDate": "2021-11-09T17:32:57.456+00:00"

6  },

7  {id: 2,

8  "chassisNumber": "CH-879763536",

9  "registrationNumber": "RG-TG089765",

10  "registrationStartDate": "2021-11-09T17:32:57.498+00:00"

11  },

12  }

13  }

14  }