

Governors State University

OPUS Open Portal to University Scholarship

All Capstone Projects

Student Capstone Projects

Fall 2022

Rent-A-Car

Shahbaaz Mohammed
Governors State University

Follow this and additional works at: <https://opus.govst.edu/capstones>

Recommended Citation

Mohammed, Shahbaaz, "Rent-A-Car" (2022). *All Capstone Projects*. 565.
<https://opus.govst.edu/capstones/565>

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the [Governors State Computer Science Department](#)

This Capstone Project is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.

RENT-A-CAR

By

Shahbaaz Mohammed

B.E., Osmania University, 2019

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science



Governors State University
University Park, IL 60484

2022

ABSTRACT

The Rent-a-Car application will improve the efficiency of the web-based Rental Car business. A renter can reserve a car from anywhere in the world. Renters from all around the world will be able to book cars through our website. The owner enters their vehicle information, which is shown on the main page. A renter who has registered on the website has the option of reserving the car they require. The renter pays the owner for the car rental and can reserve it whenever they want. The money will be deposited into the car owner's account. The vehicle's condition and details have been approved by the administrator. This automatic method assists the renter and owner by filling out the information based on their requirements.

Our application has three consoles. Owner, Admin, and Renters The administrator can manage both the owner and the renter. Owners can only manage their vehicles, bookings, and so on. Renters can view their profile, booking history, and other information. We are using .NET Core logic in Visual Studio to create the application that will handle the functionalities.

Table of Content

1	<i>Project Description</i>	1
1.1	Competitive Information	1
1.2	Relationship to Other Applications/Projects	1
1.3	Assumptions and Dependencies	1
1.4	Future Enhancements.....	2
1.5	Definitions and Acronyms	2
2	<i>Project Technical Description</i>	2
2.1	Application Architecture	3
2.2	Application Information flows.....	4
2.3	Interactions with other Applications	5
2.4	Capabilities	5
2.5	Risk Assessment and Management.....	5
3	<i>Project Requirements</i>	5
3.1	Identification of Requirements	5
3.2	Operations, Administration, Maintenance and Provisioning (OAM&P)	5
3.3	Security and Fraud Prevention.....	5
3.4	Release and Transition Plan.....	6
4	<i>Project Design Description</i>	6
5	<i>Internal/external Interface Impacts and Specification</i>	7
6	<i>Design Units Impacts</i>	11
6.1	Functional Area A/Design Unit A	11
6.1.1	<i>Functional Overview</i>	11
6.1.2	<i>Impacts</i>	11
6.1.3	<i>Requirements</i>	11
7	<i>Open Issues</i>	12
8	<i>Acknowledgements</i>	12
9	<i>References</i>	12
10	<i>Appendices</i>	12

1 Project Description

Rent a car, a company that specializes in renting cars to clients, will leverage this initiative. Customers may see available vehicles using an online system, register, view profiles, and reserve cars through it. To reserve a car, the user must first log in. The user may quickly search for cars and book ahead. The user must provide information for bookings, including the booking dates and text message. All the car's information is given, along with an overview and its features. The user may change their profile and passwords at any time on the website, as well as give comments. The administrator may add or manage car brands, manage cars, reservations, reviews, pages, and much more. Users find it simple to use and comprehend. Customers may hire cars easily with this website. The user will not have any trouble understanding, using, or navigating the design because it is rather straightforward.

1.1 Competitive Information

Additional automobile rental firms are direct rivals. Customers can buy from other businesses known as indirect rivals if they like. One of the greatest e-commerce sites for renting a car offers a variety of services for this purpose. Turo, get around, and other automobile rental companies appear to be competitors of this company. The "rent a car" website could be more dependable in this cutthroat industry by offering a location to make a booking. More trustworthy and authentic services are offered by the website. As a result, in this industry that is very competitive, this company has devoted customers and has grown its reputation(Zhongyou).

1.2 Relationship to Other Applications/Projects

The goal of this project on the car rental system is to help the rental car business make it possible to hire cars online. It enables users to look up available vehicles, see profiles, and reserve vehicles for a specific time frame. Its user-friendly design makes it easy for users to search for and rent cars for the required duration. Additionally, they may pay online. The rental vehicles must be divided into categories like budget and luxury. The user will be able to make reservations based on the kind of car the client needs. Customers may now hire a car at any time thanks to the usage of internet technologies. The reservations are simple using our car rental system(Easy Tour International Ltd). Time and work are saved. The program will request information from the user, such as the date and time of the trip, the type of vehicle, etc. Additionally, it needs a unique identifying number. This information will be used by the program to assist the user in booking a vehicle for the trip(TravelJig Saw Ltd).

1.3 Assumptions and Dependencies

In the prior system, information was manually kept on paper, and it was expensive to communicate information among personnel. However, a new solution was suggested to address the issues. The suggested system's advantages and features include:

- The sharing issue has been solved thanks to the centralization of data.
- Since data is kept in an electronic format, updating the information is simple.
- Performance is good, and maintenance is simple.
- Technology has automated the reservation and travel processes.

1.4 Future Enhancements

- ✓ In the future, the application may have a payment option, and the user might receive a billing receipt.
- ✓ The program may have an SMS alert component to alert the user. The messages have the ability to notify the user.
- ✓ To track down specific cars, vehicle tracking systems can be used.
- ✓ Users of online car rentals may receive support while driving.
- ✓ Any questions can be answered by the application's customer service line.
- ✓ Online cancellation is a feature of the program.
- ✓ For a better user experience, the program may be customized to support many languages.

1.5 Definitions and Acronyms

- ✓ HTML5 is Hypertext Markup Language 5
- ✓ CSS3 is Cascading Style Sheet
- ✓ Bootstrap is the most widely used CSS Framework for creating responsive and mobile-first websites. Bootstrap 5 is the most recent version of the framework.
- ✓ JSON is JavaScript Object Notation.
- ✓ .NET Core is the new web framework from Microsoft. .NET Core is the framework you want to use for web development with .NET
- ✓ EF is the Entity Framework Core
- ✓ .NET is Network Enabled Technology
- ✓ MVC is Model –View – Controller
- ✓ LINQ – Language Integrated Query

2 Project Technical Description

Customers will be able to rent cars from everywhere in the city thanks to the development of the car rental system. Customers provide information to this application by entering their password, email, and cellphone number. A consumer who has registered on the website has the option to reserve the car he needs. The system that is being suggested is an entirely integrated online system. It effectively and efficiently automates laborious processes. This automatic system helps the consumer and offers to fill in the information in accordance with their needs. It contains a representation of the various vehicle kinds they are attempting to hire as well as the location. The system's goal is to create a website for those who can rent cars.

Technically speaking, the project is facilitating online booking. The.NET CORE is being used by the developers on Visual Studio. Aside from that, users have access to technologies like HTML, CSS, AJAX, jQuery, LINQ, and JavaScript. The application's database is incredibly robust and secure. They store a huge number of users in the MS-SQL database. A website with many page possibilities and a well-designed database was created by the developer (R. Felder and B. Soloman, 2018). In order to have a correct workflow of the booking method for this e-commerce, the user must register them through the registration page and log in before they can book the vehicle. We utilize ASP.NET CORE Identity & Role for user sign-up and authentication. We handle user information with the use of Identity

and cutting-edge technologies like "password hashing" and "token-based password reset." The web site's authorization is handled by the ASP Role provider. We made the page access using the ASP Role. If the owner role and the renter role each have a certain function, admin. For context-based activities, we combine Lambda Expression with LINQ Statement. The code is shorter, more helpful, and simpler to understand. We also included interface features to make it simpler for the developer to enhance the program in the future.

The membership system for user authentication and authorization while developing an ASP.NET application is called ASP.NET Identity. When creating cutting-edge apps for the online, mobile devices, or tablets, The ASP.NET Identity offers a new perspective on what the membership system ought to be. You can easily alter the information about the logged-in user by adding customized login/logout functionality and customizable profile features using ASP.NET Identity. Today's online applications have access to a wider variety of data storage alternatives, and most developers want to make it possible for their websites to use social identity providers for authentication and authorization. The server uses authentication to ascertain who is accessing their data or website. The user or client must authenticate their identity on a web server by logging in using their email address and password, or one of several social media platforms. After successful authentication, a server will utilize the authorization procedure to determine if the client has permission to use a resource or access a file. Below, we will talk about the in-depth page descriptions and the database description with the application workflow.

2.1 Application Architecture

We updated the database and changed the data model in this application using the EF migrations functionality. This project began with the main page, which offers any user the standard website view of recently highlighted automobiles and a list of vehicles (guest, renter, owner, and admin) For ease of access, the highlighted automobiles list is shown on the home page. Vehicles may be found using the location, name, type, and manufacturer fields. If the renter wishes to reserve the automobile, they may do so directly on that page. If the vehicle is reserved by the renter, the owner and admin are notified. The chat feature may be used to have conversations between owners.

The Rent a Car Ecommerce Web Application's architectural diagram is shown below.

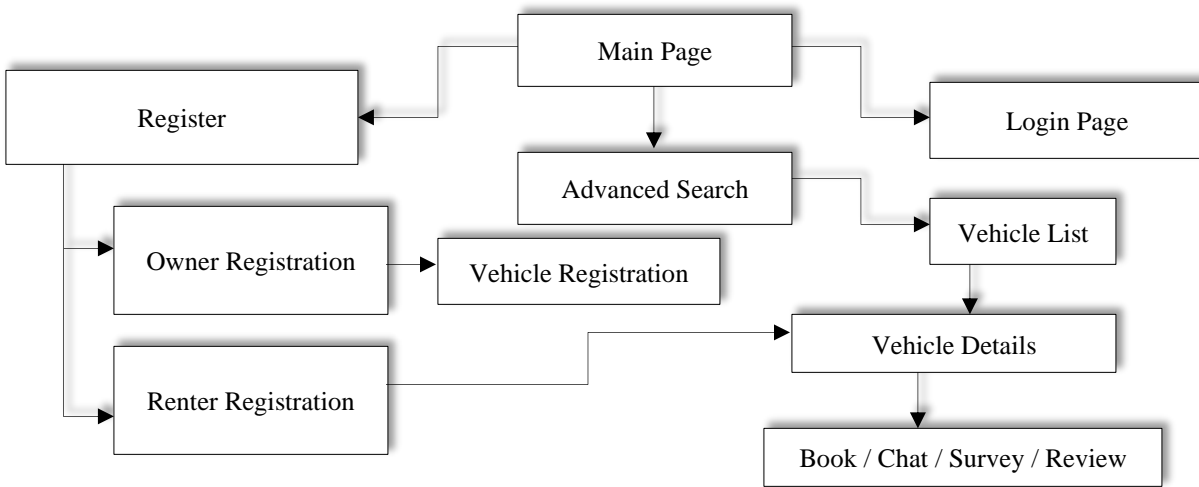


Figure 1 Application Architecture

2.2 Application Information flows

Listed below is the application workflow. The application's primary workflow is the information flow. Users may need to take the lead on the index page. The search tool and featured highlighted vehicles are included on the Index Page itself. If they choose anyone, a page with a list of vehicles and their corresponding results will appear. If the register button is clicked on the index page, the registration procedure is then opened. Once registered, they will receive admin approval and be sent to the login page. User and donor logins take them to their respective console home pages.

The conversation, booking procedure, payment, review, and survey for the car are all accessible after logging in as a renter. We place the order after receiving the payment information. The booking history includes a list of all booked vehicles. Owners may access car registration, renter chat, booking history, and payment information by logging in as themselves. As soon as they log in as the administrator, users may access user lists, vehicle management, user approval, and user lists that offer information about all currently logged-in users so that they can change their profiles. The administrator can add, edit, and delete information regarding car type and manufacturer. All users may change and update their information and password by clicking their profile, which takes them to my profile page. To keep their information secure, all users must successfully log out of the website at the conclusion of the procedure.

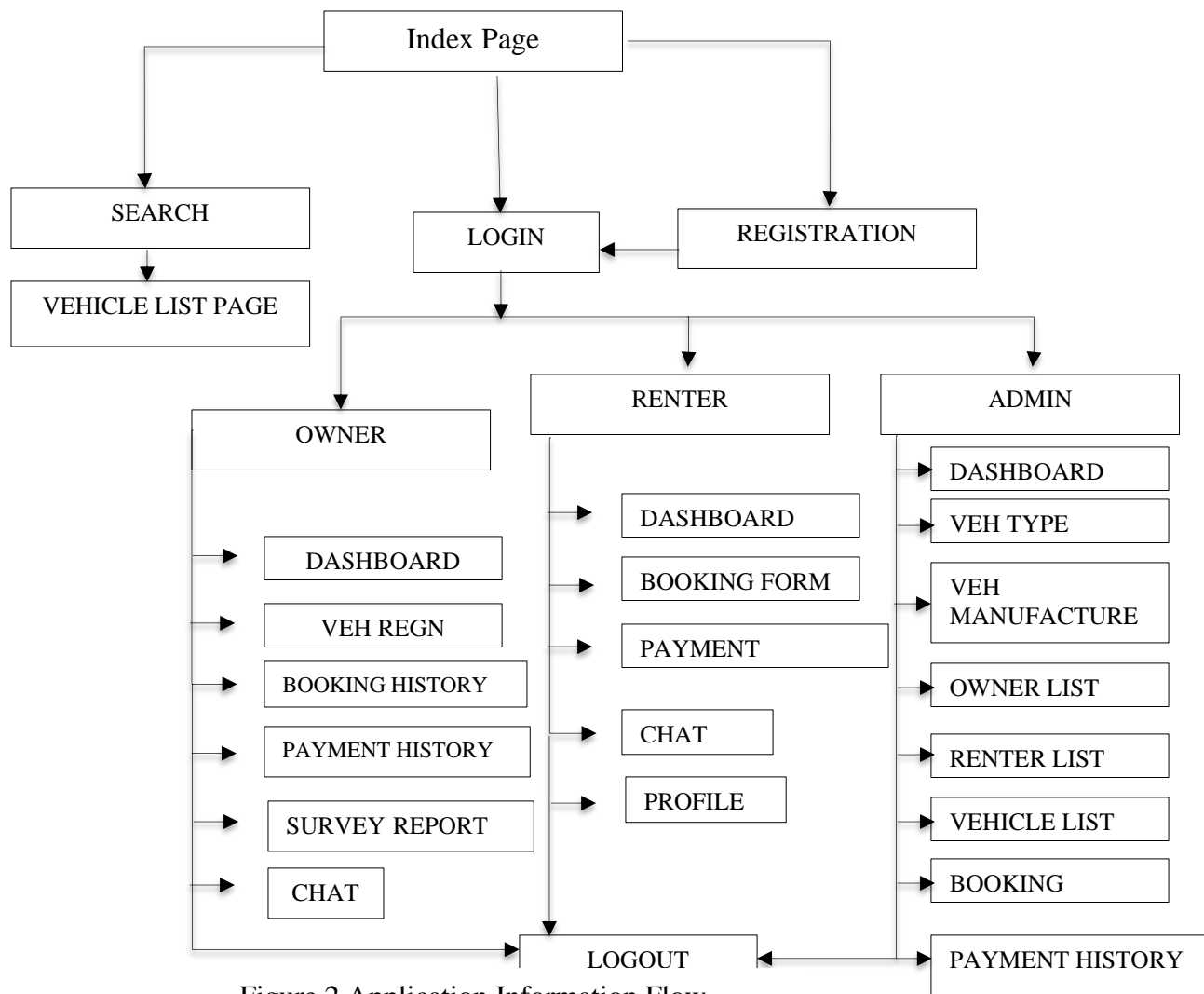


Figure 2 Application Information Flow

2.3 *Interactions with other Applications*

The benchmark of the rival firm was to evaluate the system's effectiveness and based on the researcher's collected customer experience, to suggest a better platform for the case company. The competitive landscape was examined by renting a car. The renting procedure is simple to use and very engaging, according to the competitor's study. The rival business likewise promotes extra services through the car rental user interface. Additionally, the pricing is appealing and affordable when compared to other traditional car rental businesses, giving them a significant competitive edge.

2.4 *Capabilities*

- ✓ We should include sophisticated search criteria to make booking easier.
- ✓ Rentals based on location
- ✓ listing and information on vehicles
- ✓ Allow communication between the renter and owner

2.5 *Risk Assessment and Management*

Assessment of Potential Clients: You have the right to evaluate potential clients before any transactions. You should not conduct too much research since doing so will make potential consumers uneasy.

Well-Organized Rental Administration: In the rental industry, excessive risk is frequently a result of poor administration. You need capable admins for that. Your company becomes more effective and efficient with good governance.

Provide a dedicated budget for car tracking devices to safeguard your company's assets. Installing car tracking systems is one technique to keep an eye on your rental cars.

3 *Project Requirements*

3.1 *Identification of Requirements*

<RENTACAR-RC_2022-01 User-capability-000100>

The application's project needs a user-friendly interface and a straightforward architecture.

< RENTACAR-RC _2022-02 Renter-capability-000101>

Only the renter is permitted to submit a review or survey; everyone else is unable to do so.

< RENTACAR-RC _2022-03 Owner-capability-000102>

Only the owner may register the car.

< RENTACAR-RC _2022-04 Admin-capability-000103>

The owner, renter, and vehicle may be managed by admin only.

3.2 *Operations, Administration, Maintenance and Provisioning (OAM&P)*

The user data is safely stored in a SQL Database that is frequently updated and will be closely watched by the administrator staff. The programme gives registered and legitimate users access to safe transactions. For access to the console, registration and login should be required. Customers who are not registered can examine the list of vehicles and car data. The search process needs to go quickly and realistically. The feature category list should be displayed on the website.

3.3 *Security and Fraud Prevention*

- ✓ Rental cars adhere to fundamental safety and environmental standards.

- ✓ The car needs to be legitimately registered in the operating nation.
- ✓ The condition should be in decent shape inside and out and be quite recent.
- ✓ It must verify with operational needs (for example, four-wheel drive for off-road use for a city).
- ✓ It should be equipped with the required tools and equipment.
- ✓ It must possess all necessary documents, including permission to go to all the places where the operation calls for it (technical passport for the vehicle, local insurance, pollution control certificate, valid yearly road tax payment, etc.)
- ✓ Rental agreements must specify the days and hours of the rental, specify that all necessary repairs and maintenance will be made at the owner's expense, and stipulate that there will be one paid maintenance day per month.

3.4 Release and Transition Plan

Website development cannot start until the site is finished and extensively tested. The rental car website must be published before the issues are fixed. After booting, the data storage process will start, and the maintenance procedure will be used to maintain the application's continuous operations.

4 Project Design Description

Dot Net Core [EF Core 6] was used to build every aspect of this project. We used the MVC pattern format to do this. We picked the webpage as a "getaround" by imitating it (Speed Rent Technology). In our application, we used two different layouts: one for the main menu and another for registered users. The layout's primary objective is to provide all types of car information, both with and without a registration process. If a person updates their profile, we just utilize their legitimate username to update it. For all user categories, we used a single registration. We utilized a single login page for all users, including administrators, renters, and owners, who all used the same login page. We allow each user to take on their own duties, not those of other people. For instance, a tenant cannot act in the capacities of an administrator or an owner.

When a user logs in as an administrator, the Admin Layout Page appears as the first page. The administrator may view all the data associated with this application on the Dashboard using this Admin Layout. Only the administrator has the authority to decide on vehicle type, manufacturer, owner, renter, and management of the cars. The booking and payment history is seen by the admin. A user is immediately sent to the owner layout page after logging in. We also show some recent activity here, such as the number of postings a particular owner's vehicles have had to this point, the number of vehicles that the renter has reserved, and the number of chats that have reached the renter. When a renter logs in, they are directed to the renter layout page. Renters may change their personal information on the dashboard, including how many cars they have reserved. You may look for a car list using the search bar on the layout page. Each of them will be able to view information on each car. Renters who want to reserve a car must first register as renters, and owners who want to list a vehicle for their business must first register as owners.

5 *Internal/external Interface Impacts and Specification*

The defining of requirements is a crucial phase in the life cycle of software development. Although a number of software and hardware specifications may be used to produce this application, the software and hardware requirements that we used to develop this job search website are stated below:

Software Requirements:

Operating System	:	Windows 10
Application	:	Visual Studio C#
Database	:	MSSQL
Front End	:	HTML5, CSS3, Bootstrap5, JavaScript
Browser	:	Preferable Google Chrome or Mozilla Firefox or Edge

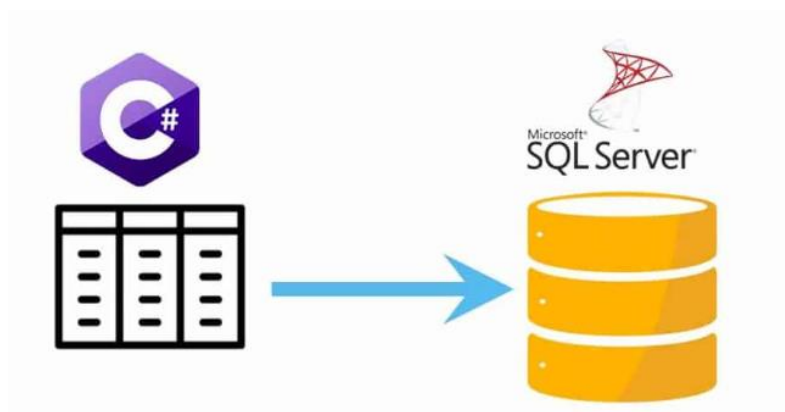


Figure 3: Internal Database and Backend

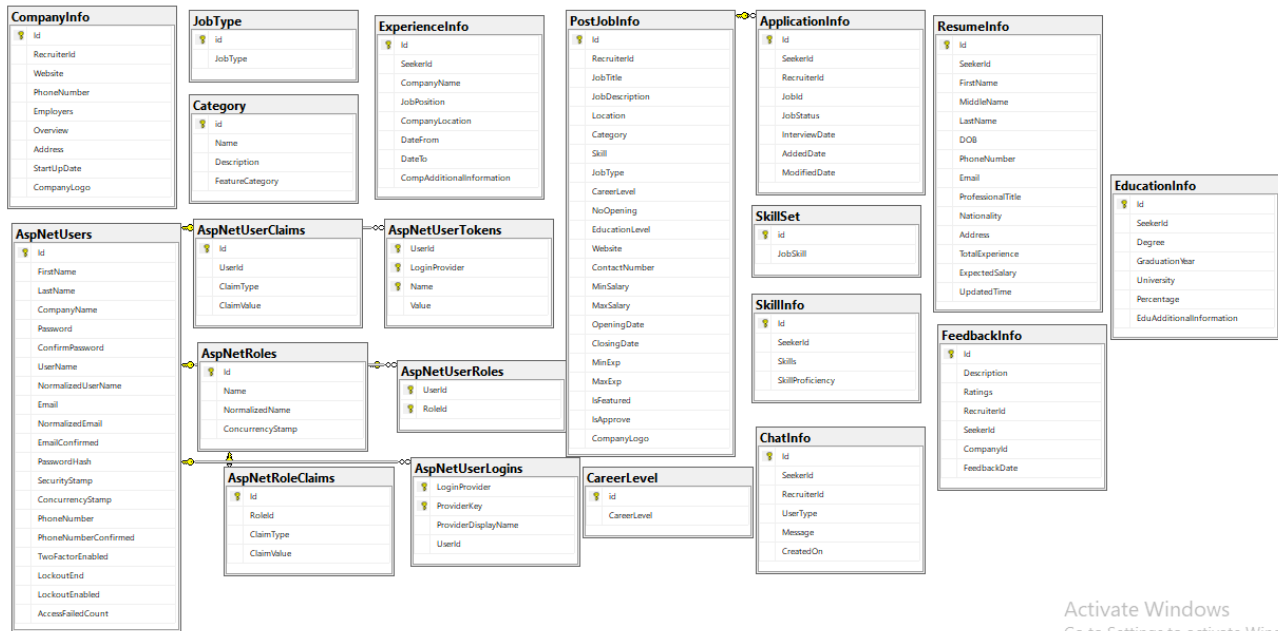


Figure 4: Database ER Diagram

The aspects of the application's design may be seen in the following screenshots.:

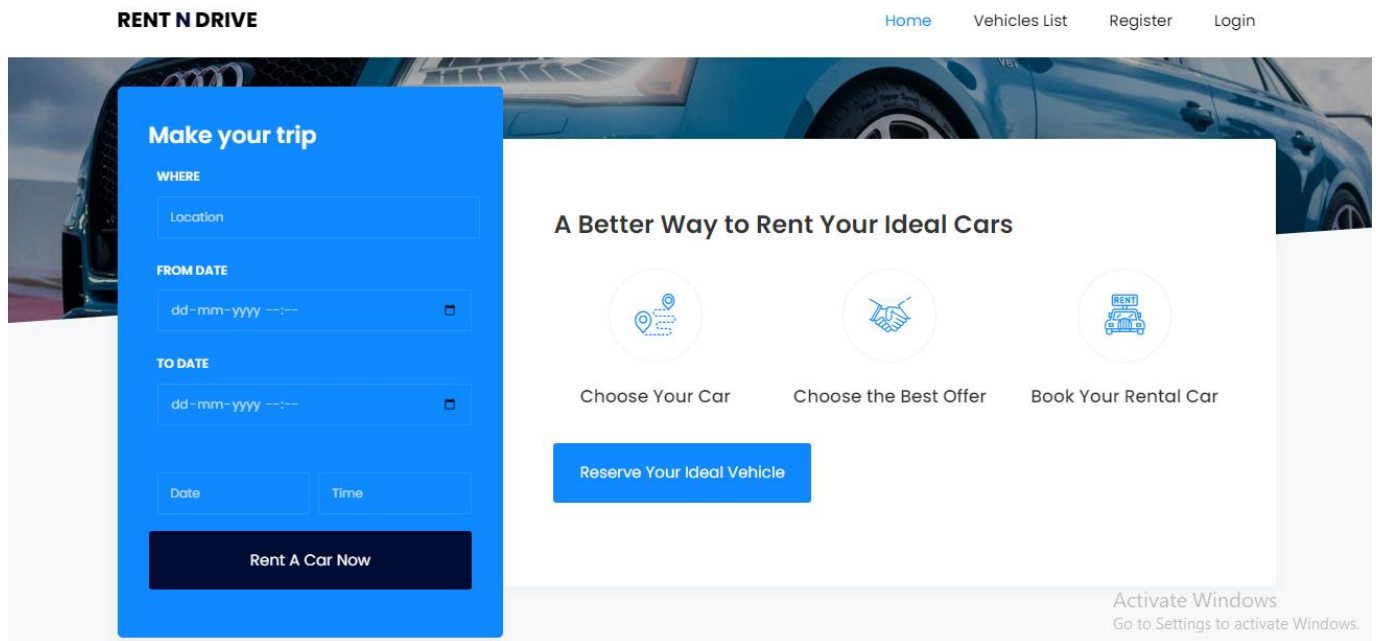


Figure 5 Home Page

SPORTSCAR | TATA MOTORS | VOWNER

Tata Suzuki

TN211123 | 20-05-2002 00:00:00 | CH



Hourly
\$ 100.00



Transmission
Automatic



Seats
2



Luggage
2 Bags

[Book now](#) [Chat now](#)

[Features](#) [Terms and Conditions](#) [Review](#) [Survey](#)

- ✓ Dual Zone Air Conditions
- ✓ Yellow Colour
- ✓ 2 Doors

Figure 6 Vehicle Details Page

General Information

First Name

Last Name

Owner ▼

Role

Owner Name

Contact Details

PhoneNumber

Email

UserName Password

RePassword

I do accept the [Terms and Conditions of your site.](#) [Settings to ac](#)

Figure 7 Registration Page



Login Form

Email address

Password

Remember me
 Forgot password?

Don't have an account? [Register](#)

Figure 8 Login Page

Rent N Drive

Search
🔍

Admin

MAIN

- 🏠 Dashboard
- 🚗 Vehicle Type
- 🔧 Vehicle Manufacturer
- 👤 Owner List
- 📄 Renter List
- 🚗 Vehicle List
- 📅 Booking History
- 💰 Payment History

<div style="font-size: 2em; font-weight: bold;">1</div> <div style="font-size: x-small;">OWNER</div>	<div style="font-size: 2em; font-weight: bold;">1</div> <div style="font-size: x-small;">RENTER</div>	<div style="font-size: 2em; font-weight: bold;">5</div> <div style="font-size: x-small;">VEHICLE TYPE</div>	<div style="font-size: 2em; font-weight: bold;">1</div> <div style="font-size: x-small;">MANUFACTURER</div>
<div style="font-size: 2em; font-weight: bold;">2</div> <div style="font-size: x-small;">VEHICLES COUNT</div>	<div style="font-size: 2em; font-weight: bold;">1</div> <div style="font-size: x-small;">BOOKING HISTORY COUNT</div>	<div style="font-size: 2em; font-weight: bold;">2</div> <div style="font-size: x-small;">CHAT COUNT</div>	

Figure 9 Admin Layout

Rent N Drive

Search
🔍

vo@gmail.com

MAIN

- 🏠 Dashboard
- 📄 Vehicle Registration
- 📅 Booking History
- 💰 Payment History
- 🗨️ Survey Report
- 🗨️ Chat

<div style="font-size: 2em; font-weight: bold;">2</div> <div style="font-size: x-small;">VEHICLES COUNT</div>	<div style="font-size: 2em; font-weight: bold;">1</div> <div style="font-size: x-small;">BOOKING HISTORY COUNT</div>	<div style="font-size: 2em; font-weight: bold;">2</div> <div style="font-size: x-small;">CHAT COUNT</div>
---	--	---

Figure 10 Owner Layout

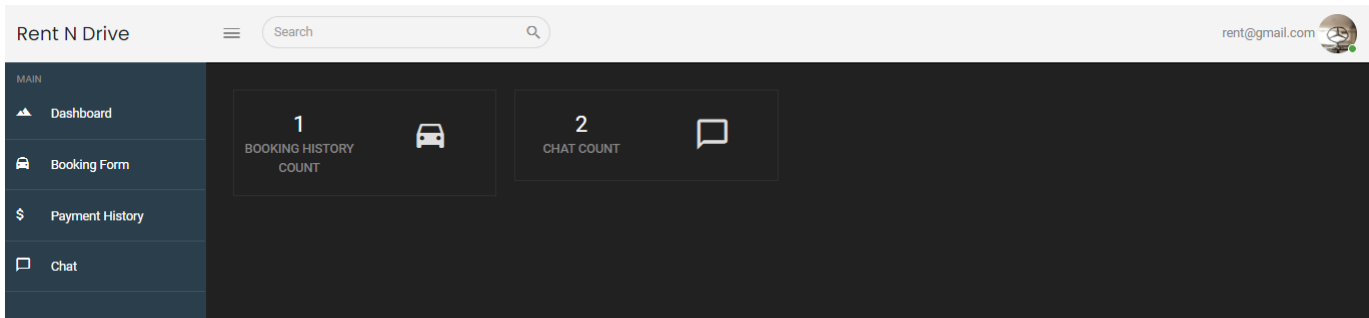


Figure 11 Renter Layout

6 Design Units Impacts

It became clear that the new car rental platform makes the process simpler and easier for clients based on the results of the surveys, competitor service offerings, literature, and best practice discoveries. The most evident among the other possible factors for this system's convenience is the time and money saved. Additionally, the new system offers flexible service hours and availability throughout the day and night, both of which were not feasible with the previous auto rental system.

6.1 Functional Area A/Design Unit A

6.1.1 Functional Overview

Requirement analysis is a software engineering approach made up of many activities that identify the criteria that must be satisfied for a new or modified product while taking into account any potential conflicts between the requirements of different customers. Functional requirements are those that serve as examples for the system's internal operation, its description, and an explanation of each subsystem. It includes the task the system should do, the procedures involved, the data the system should retain, and the user interfaces.

6.1.2 Impacts

This design unit and workflow process, the computer's operating functions may modify the system to meet operational and customer demands. The work process includes authentication, data updates, user information updates, backup and recovery procedures, and administrative operations.

6.1.3 Requirements

- ✓ The system will include a customer portal website that will inform the public and consumers about the business and how it operates.
- ✓ The System will make it possible for the company's inventory to grow by adding additional rent the cars.
- ✓ The user may enter their preferred date and select choices to find out which vehicles are available using the system. The system will proceed to the next phase if a car is available. The system will ask the consumer to choose another car if the requested vehicle is not available.
- ✓ Name, email address, and phone number are needed for the customer and renter account.

7 Open Issues

No Issues

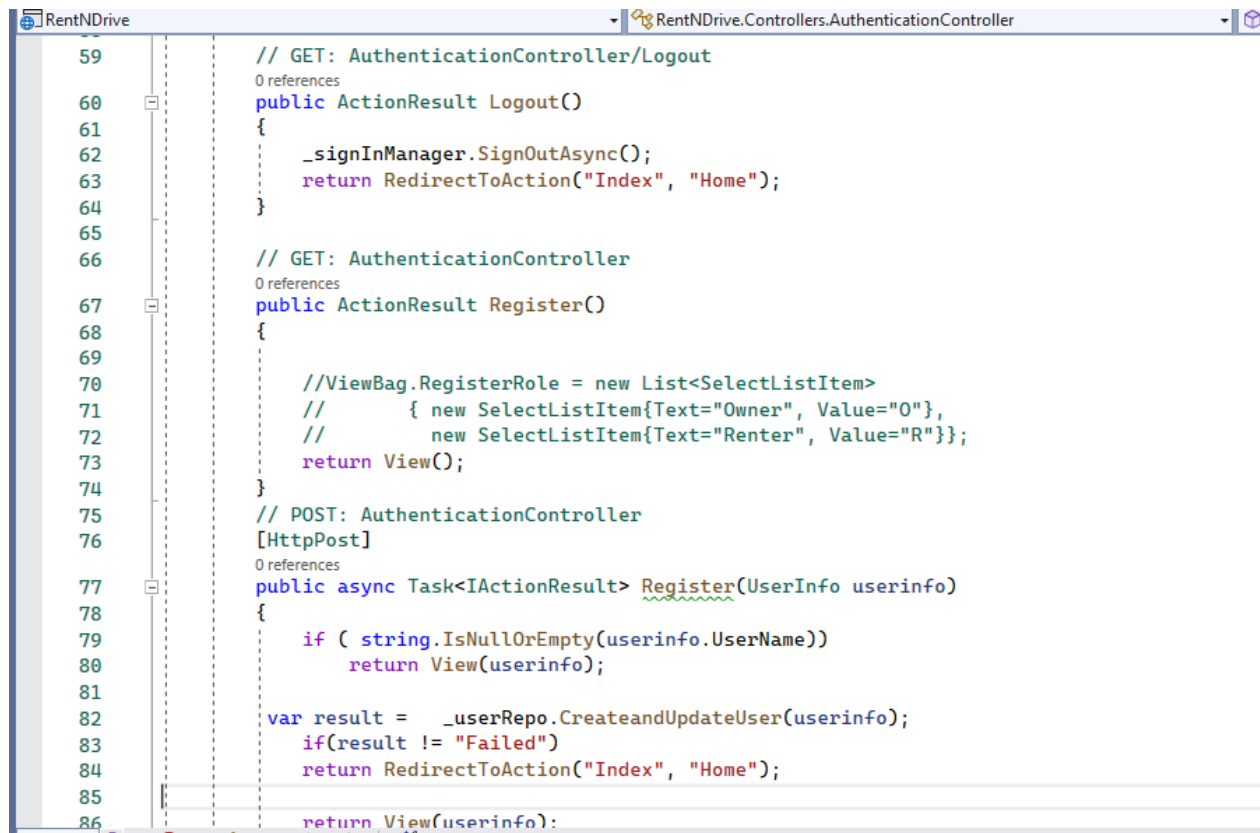
8 Acknowledgements

I would like to thank my professor (Dr. Xin Chen) for giving me the excellent opportunities to create this wonderful project plan. I was able to perform significant study because of this initiative. I will learn a lot from this rental vehicle project. It helped me develop my research and report-writing skills. My teammates helped me do the job successfully, therefore I would like to thank them as well.

9 References

- [1] Zhongyou, X. (2012). Study on the Application of TOPSIS Method to the Introduction of Foreign Players in CBA Games. SciVerse ScienceDirect. <https://cyberleninka.org/article/n/152325>.
- [2] Easy Tour International Ltd © easyrentcars.com. Retrieve from <https://www.easyrentcars.com/>
- [3] TravelJigsaw Ltd © 2019 TravelJigsaw Limited (2004). Retrieve from <https://www.rentalcars.com/>
- [4] Speedrent Technology Sdn Bhd (2015). Retrieve from <https://speedhome.com/>

10 Appendices



```
59 // GET: AuthenticationController/Logout
60 0 references
61 public ActionResult Logout()
62 {
63     _signInManager.SignOutAsync();
64     return RedirectToAction("Index", "Home");
65 }
66 // GET: AuthenticationController
67 0 references
68 public ActionResult Register()
69 {
70     //ViewBag.RegisterRole = new List<SelectListItem>
71     //     { new SelectListItem{Text="Owner", Value="O"},
72     //       new SelectListItem{Text="Renter", Value="R"}};
73     return View();
74 }
75 // POST: AuthenticationController
76 [HttpPost]
77 0 references
78 public async Task<IActionResult> Register(UserInfo userinfo)
79 {
80     if (string.IsNullOrEmpty(userinfo.UserName))
81         return View(userinfo);
82     var result = _userRepo.CreateandUpdateUser(userinfo);
83     if(result != "Failed")
84         return RedirectToAction("Index", "Home");
85
86     return View(userinfo);
```

Figure 12 Authentication Controller


```

RentNDrive | RentNDrive.Controllers.AdminController
59     }
60     // GET: AdminController
61     [HttpGet]
62     0 references
63     public IActionResult carType(int? id)
64     {
65         var vehType = new VehTypeView();
66         vehType.vehicleTypeList = _vehTypeR.GetList();
67         var idValue = id > 0 ? vehType.vehicleTypeNew = _vehTypeR.GetListById((int)id) :
68         return View(vehType);
69     }
70     // POST: AdminController
71     [HttpPost]
72     0 references
73     public IActionResult carType(int id, VehTypeView vehType)
74     {
75         if (vehType.vehicleTypeNew == null)
76         {
77             return View(vehType);
78         }
79         var result = _vehTypeR.CreateAndUpdateVType(id, vehType.vehicleTypeNew);
80         if (result == "Success")
81         { return RedirectToAction("carType"); }
82         return View(vehType);
83     }
84     0 references
85     public ActionResult Cancel()
86     {
87         var vehicleTypeClear = new VehicleType();
88         return RedirectToAction("carType");
89     }

```

Figure 13 Admin Controller

```

RentNDrive | RentNDrive.Controllers.AdminController
117     [HttpGet]
118     0 references
119     public async Task<IActionResult> OwnerList(AdminView adminView, string uservalue)
120     {
121         ViewBag.uservalue = uservalue;
122         adminView.UserList = uservalue == "Owner"? _userR.GetAllUserList().Where(o=>o.Ow
123         return View("UsersList", adminView);
124     }
125     0 references
126     public IActionResult ActiveInactive(string? id, int? vehicleId)
127     {
128         var status = _userR.GetUserById(id); // _context.Users.Where(x => x.Id == id).ToLi
129         if (status != null)
130         {
131             status.EmailConfirmed = !status.EmailConfirmed;
132             _context.Entry(status).State = EntityState.Modified;
133             // _context.Users.Update(status);
134             _context.SaveChanges();
135             var userval = status.OwnerName != null ? "Owner" : "Renter";
136             return RedirectToAction("OwnerList", new { uservalue = userval });
137         }
138         var vehiclestatus = _vehicleR.GetListById((int)vehicleId);
139         if (vehiclestatus != null)
140         {
141             vehiclestatus.VehicleActiveStatus = !vehiclestatus.VehicleActiveStatus;
142             _context.Entry(vehiclestatus).State = EntityState.Modified;
143             _context.SaveChanges();
144             return RedirectToAction("VehicleRegistration", "Owner");
145         }
146         return RedirectToAction("Dashboard");

```

Figure 14 Vehicle Active / Inactive

```

RentNDrive - RentNDrive.Controllers.HomeController
28
29 0 references
30 public HomeController(UserManager<UserInfo> userManager,
31     SignInManager<UserInfo> signInManager, RoleManager<IdentityRole> roleManager,
32     ApplicationDbContext context, IUser userRepos, IVehicle vehicleRepos, IVehManufactur
33 )
34 {
35     0 references
36     public IActionResult Index(VehicleView vehicleViewModel)
37     {
38         vehicleViewModel.GetAllVehicles = _vehicleRepo.GetList().Where(x => x.FeatureVehicle
39         return View(vehicleViewModel);
40     }
41     0 references
42     public IActionResult VehicleList(VehicleView vehicleViewModel)
43     {
44         vehicleViewModel.GetAllVehicles = _vehicleRepo.GetList().Where(x => x.VehicleActive!
45         vehicleViewModel.GetAllVehicles.ForEach(v =>
46         {
47             v.NMVehicleManufacturer = _vehmanRepo.GetListById(v.VehicleManufacturer).Vehicle
48             v.NMVehicleType = _vehtypeRepo.GetListById(v.VehicleType).VehicleTypeName;
49             v.NMOwnerName = _userRepo.GetUserById(v.OwnerId).OwnerName;
50         });
51         return View(vehicleViewModel);
52     }
53     0 references
54     public IActionResult VehicleDetails(VehicleView vehicleViewModel, int? id)
55     {
56         vehicleViewModel.GetVehicleInfo = _vehicleRepo.GetListById((int)id);
57         vehicleViewModel.GetAllVehicles = _vehicleRepo.GetList();
58         vehicleViewModel.GetAllVehicles.ForEach(v =>
59         {
60             v.NMVehicleManufacturer = _vehmanRepo.GetListById(v.VehicleManufacturer).Vehicle
61

```

Figure 15 Home Controller

```

RentNDrive - RentNDrive.Controllers.OwnerController - VehicleRegistration(int? id, VehicleView vehicleView)
67
68 // GET: Renter/Create
69 0 references
70 public IActionResult VehicleRegistration(VehicleView vehicleView, int? id)
71 {
72     var CurrentUserId= this.User.FindFirstValue(ClaimTypes.NameIdentifier);
73     ViewBag.VehicleType = _context.vehicleType.Select(x => new SelectListItem { Text = x.VehicleTypeName, Value = x
74     ViewBag.VehicleManufacturer = _context.vehicleManufacturer.Select(x => new SelectListItem { Text = x.VehicleMan
75     vehicleView.GetAllVehicles = User.IsInRole("Owner")? _vehicleR.GetList().Where(x => x.OwnerId == CurrentUserId
76     _vehicleR.GetList().ToList();
77
78     if (vehicleView.GetAllVehicles.Count > 0)
79     {
80         foreach (var item in vehicleView.GetAllVehicles)
81         {
82             item.NMVehicleManufacturer = _vehmanR.GetListById(item.VehicleManufacturer).VehicleManufacturerName; //
83             item.NMVehicleType = _vehtypeR.GetListById(item.VehicleType).VehicleTypeName;
84         }
85         return View(vehicleView);
86     }
87     return View();
88 }
89

```

Figure 16 Owner Controller

```

public RenterController(UserManager<UserInfo> userManager, SignInManager<UserInfo> signInManager,
    RoleManager<IdentityRole> roleManager, IUser userR, ApplicationDbContext context, IVehType vehTypeR, IVehManufacturer vehman)
0 references
public IActionResult BookNow(VehicleView vehicleViewModel, int vehId)
{
    var CurrentUserId = this.User.FindFirstValue(ClaimTypes.NameIdentifier);
    vehicleViewModel.GetAllBookingHistory = User.IsInRole("Renter") ? _vehicleR.GetBookList().Where(x => x.RenterId == CurrentUserId)
        : User.IsInRole("Admin") ? _vehicleR.GetBookList().ToList() : _vehicleR.GetBookList().ToList();
    if (CurrentUserId != null && User?.Identity?.IsAuthenticated == true)
    {
        vehicleViewModel.GetVehicleInfo = _vehicleR.GetListById(vehId);
        vehicleViewModel.GetUserInfo = _userR.GetUserById(CurrentUserId);
        vehicleViewModel.GetAllBookingHistory.ForEach(v =>
        {
            var ownerId = _vehicleR.GetListById(v.VehicleId).OwnerId;
            v.NMOwnerName = _userR.GetUserById(ownerId).OwnerName;
            v.NMVehicleName = _vehicleR.GetListById(v.VehicleId).VehicleName;
            v.NMRenterName = _userR.GetUserById(v.RenterId).FirstName;
        });
        return View("BookInfo", vehicleViewModel);
    }
    return RedirectToAction("Login", "Authentication");
}

```

Figure 17 Renter Controller

```

RentNDrive
RentNDrive.Models.Chat
28 public decimal VehiclePrice { get; set; }
29 [Required, Display(Name = "Price Type (eg. Hour, Week)")]
18 references
30 public string PriceType { get; set; }
31 [Display(Name = "Doors")]
15 references
32 public int Doors { get; set; }
33 [Display(Name = "Colour")]
6 references
34 public string Colour { get; set; }
35 [Display(Name = "Passengers")]
15 references
36 public int Passengers { get; set; }
37 [Display(Name = "Luggage")]
14 references
38 public string Luggage { get; set; }
39 [Display(Name = "Transmission")]
14 references
40 public string Transmission { get; set; }
41 [Display(Name = "Air conditioning")]
14 references
42 public string AirConditioning { get; set; }
43 [Display(Name = "Terms")]
12 references
44 public string Terms { get; set; }
45 [Display(Name = "Vehicle Available Status")]
8 references
46 public bool VehicleAvailableStatus { get; set; }
47 [Display(Name = "FeatureVehicle")]
14 references
48 public bool FeatureVehicle { get; set; }
49 [Display(Name = "Vehicle Images")]
13 references
50 public string VehicleImage { get; set; }
51 [Display(Name = "Vehicle Show / Hide")]
14 references

```

Figure 18 Vehicle Model

```

RentNDRive |> RentNDRive.ImplementationLogics.IBusinessLogic.IVehType
1  using RentNDRive.Models;
2
3  namespace RentNDRive.ImplementationLogics.IBusinessLogic
4  {
5      public interface IVehType
6      {
7          string CreateandUpdateVType(int Id, VehicleType vehicletypename);
8          VehicleType GetListById(int Id);
9          List<VehicleType> GetList();
10     }
11
12     public interface IVehManufacturer
13     {
14         string CreateandUpdateVManufacturer(int Id, VehicleManufacturer vehiclemanufacturer);
15         VehicleManufacturer GetListById(int Id);
16         List<VehicleManufacturer> GetList();
17     }
18 }
19

```

Figure 20 Interface

```

RentNDRive |> RentNDRive.ImplementationLogics.BusinessLogic.UserRep
19     _signInManager = signInManager;
20     _roleManager = roleManager;
21 }
22
23 public string CreateandUpdateUser(UserInfo userDetails)
24 {
25     var result = _userManager.CreateAsync(userDetail, userDetails.Password);
26     if (!result.Result.Succeeded)
27     {
28         return "Failed";
29     }
30     if (userDetail.OwnerName != null)
31     {
32         result = _userManager.AddToRoleAsync(userDetail, "Owner");
33     }
34     else
35     {
36         result = _userManager.AddToRoleAsync(userDetail, "Renter");
37     }
38     result.Wait();
39     return "New User Successfully Created";
40 }
41
42 public IEnumerable<UserInfo> GetAllUserList()
43 {
44     return (IEnumerable<UserInfo>)_con.Users.ToList();
45 }
46
47 public UserInfo GetUserById(string Id)
48 {
49     return _con.Users.FirstOrDefault(x => x.Id == Id);
50 }

```

Figure 21 Repository