# COMP1682 Final Report

**Project Name: Tourist Car Rental Application (TCAR)**

**Student Name: Tran Dinh Minh**

**Student ID: 001197133**

**Supervisor: Tran Trong Minh**

**Date Submit: 04/29/2022**

# Final Year Report

## COMP1682 Final Year Project

Program Title: BSc Hons Computing

## Abstract

Nowadays, cities in Vietnam are increasingly developing in the field of tourism. People tend to go sightseeing or entertainment after tiring working hours and the way they want to solve their stress problems after long days is a trip. When traveling with each city in Vietnam, we cannot help but mention the buses carrying many passengers with the tour guide department who know many places to visit in the city they are working to support to help passengers get to place they do not know. Because tourists are increasing and the demand for car rental is also growing, the objective of this research is to develop a tourist car rental application that has performance in many famous cities in Vietnam male. The application will let users search for a car, rent a car quickly without having to go through many steps, with the primary purpose of being safe because the driver department will be rigorously trained for tourists to tourist have the most satisfying trip. Moreover, the application will also analyze the suitability of the system based on collecting data from literature review to get the most specific and accurate numbers. The application will be developed with the full stack MERN (Mongo DB, Express, ReactJS, NodeJS) of the JavaScript language and will be tested specifically, then will find a cloud to deploy the application.



## Acknowledgements

First, I want to thank Mr. Tran Trong Minh for supporting and helping me throughout the process of studying for the final project. From the proposal, he helped me come up with ideas to develop a necessary application to serve tourists in Vietnam. The necessary knowledge and my errors in this report were all corrected by his enthusiasm for the purpose of helping me pass this course. Once again, I would like to sincerely thank Mr. Tran Trong Minh for being by my side and assisting me to complete this report.

Secondly, I would also like to thank Mr. Hoang Nhu Vinh for analyzing for me a clear understanding of the client and server model, specifically analyzing the risks that the project may face. He also supported me with the necessary functions of the project that required the support of 3rd parties to bring about the tightness of the system.

Finally, I want to thank my family for always supporting and comforting me during my final project so that I can pass this course with an important milestone to work at a certain company most passionately.

Table of Contents

## Table of Tables



## Table of Figures

# 1. Introduction:

## 1.1. Background:

In Vietnam, tourism is considered a spearhead economic sector with rich potential and future development. Thanks to the places and rich elements, the places attract a lot of domestic and international tourists. According to Nguyen Thi Hoa, from 2016-2019, the number of international visitors to Vietnam is increasing day by day with a huge number from 72 million to 103 million visitors with a high growth rate (Nguyen Thi Hoa, 2021).These are considered factors that show the hotness of the tourism industry in Vietnam.

However, according to Wikipedia, on January 23, 2020, the first positive case of COVID-19 caused by the SARS-CoV-2 virus was in Ho Chi Minh City, Vietnam. Because of covid 19, the tourism industry is also going down (Wikipedia, 2022). According to Margaux Constantin, Matthieu Francois, and Thao Le, The tourism industry declined and accounted for only 12% of the country's GDP and the number of international visitors also was only 17% in 2019, these are the numbers sad specifics of the country's tourism industry (Margaux Constantin, 2021). According to Alexandria Cahill, let is take a look at the tourism situation from 2000 to 2019, tourism developed with great numbers, from 2000 to 2013 from 1.2 million to 35 million people came to travel in Vietnam and from 2013 to 2019 increased 50 million more people, of which 17 to 20 million other people come to Vietnam from abroad (Cahill, Fall 2018).

The covid situation can't last long, and finally there is good news coming to Vietnam to save the lousy numbers of the country's tourism. According to Government News, on February 17, 2022, Vietnam covered all of nose 3 with the aim of people living with the epidemic so the demand for travel will be developed again (News, 2022). Let is take a look at the problems of previous years, the car rental tourism industry has many problems because the demand for car rental is increasing, leading to the car rental service in Vietnam sometimes overloaded. The number of people using car rental services has skyrocketed year-over-year with an annual growth of 34% and 48% based on (Ken, 2019). According to Ken, there are still many tourists who have not caught up with the trend of switching to online car booking through apps, although car rental services already exist and are owned by companies like MGM, SaiGon Star, and so on, but still have not meet the needs of tourists (Ken, 2017). To make this no longer a problem this year and in the future, I want to develop an easy-to-use and professional travel car rental application on top so that users can easily use the application with payment online, chatbot assist users in searching for cars with professionally trained drivers with the aim of avoiding unnecessary risks. Surely, the application will bring an interesting experience.

The project will focus heavily on developing user interface, performance, functions suitable for specific scheduled process clearly with real data collected from users. The tools I will be using are the full Stack MERN (Mongo DB, Express, ReactJS, NodeJS). With the above factors in mind,

I decided to develop a tourist car rental application with the aim of supporting the country's tourism industry to develop further and find a solution to overcome the unexpected accident situation for customers with more than 300 tourists agree me to develop this app. The app is called Tourist Car Rental App (TCAR).

### 1.2. Aim:

This project was built with the aim of developing an increasingly professional tourist car rental service, providing a friendly user interface and experience, minimizing overload when renting a touring car. Moreover, ensure the safety of each passenger with confidence when renting a car because the drivers will be professionally trained with the priority mindset of "Say no to risks". The main aim is to bring about the necessary profit for the company. In addition, developing this application helps customers from many places to book cars without having to go directly to TCAR company.

## 2. Objectives:

### 2.1. The research essentials the following factors:

I need to research and define software development techniques. Identify user needs for application development. Use the document review to determine the exact scope of the system. The main purpose of developing this app is to help me better understand how the front-end and back-end of an app work. In addition, understanding the standards and application scopes will help me develop the system according to the user's requirements. Moreover, surveying the needs of customers who want the application to develop through data from tourists every year in Vietnam. Investigate how successful the system has been with the initial goals of limiting overcrowding with car rental services, providing the best passenger experience, and delivering high profits for the company.

### 2.2. Collect and analyze requirements:

Collect the necessary information from national tourists or even foreign tourists. I will proceed to create a google form questionnaire to cover all the opinions from the survey participants. From these factors, I will conduct requirements analysis, then draw the left and right side for the application.

### 2.3. Research technology:

I will conduct research on web application technologies with the goal of cross-platform development. Clearly define the decentralization of the system along with security. Research which back-end or front-end frameworks are suitable for the system. Researching the database will be suitable for the car rental application, the database will need to be accurate and not messed up. Next, I also researched which 3rd parties are willing to support me in the login function with Google or Facebook to bring many good options to users. Finally, I also

researched the trend of easy online payment with the goal of ensuring the safety of each visitor when using the application's services.

## 2.4. Development:

### 2.4.1. Design for app:

- Use Case Diagram: Used to describe user interaction with application features. It makes it easy for me to identify the functions that the application needs to interact the user requirements.

- Entity Relationship Diagram: Use ERD for the purpose of representing the entities in the database, and the relationship between them. The elements that link the tables together help create the logic to connect the data to the application.

- Design and prepare content for app: I will use wireframe to develop the front-end for the project. The project will be beautifully designed with clear, eye-catching split layouts intended to not confuse users with front-end layouts. The content will be professionally written with clear and minimal text and lots of images or videos so that users feel satisfied with the front-end of the application.

### 2.4.2. Choose technical for app:

The application will be developed with the latest technologies to provide the best user experience. On the front-end side, I will use HTML, CSS, JS, ReactJS. On the back-end side, I would use NodeJS. MongoDB will be the project's database.

### 2.4.3. Built interface and server for app:

Conduct reference to UI & UX websites with beautiful layouts to cook for the best project experience. Then proceed to develop necessary functions such as registration, login, information search, payment, call a loved one if an accident happens, admin can manage lower roles in the system, decentralization, and so on.

## 2.5. Testing and deployment:

### 2.5.1. Testing:

After the application development process, I will conduct testing to see if the system is working according to the original goal. If the system encounters errors or problems arise, I will fix errors and improve the product as soon as possible to keep up with the project schedule. A final stage of testing will perform is acceptance testing, with the participation of the user in the main role to determine whether the software system meets the user's requirements or not.

#### 2.5.2. Deployment:

I will deploy the application with Heroku, Glitch, Google Cloud Platform, AWS, or Netlify based on the right platform for my system.

### 2.6. Evaluation:

Evaluate whether the functions have been completed against the original goal? Is the application functioning properly when interacting with the user? Identify the risks when the application is officially released and look for bugs that may arise. If system failures are identified, corrective action should be taken immediately. If the product has been completed and there are no problems, then evaluate what elements the application is missing, need maintenance and what needs to be improved so that the application can develop more in the future.

## 3. Approach:

For each project, there must be a SDLC model (Software Development Life Cycle) to track the development process for a software project. SDLC is a blueprint to describe how to develop, fix, maintain or upgrade software. Therefore, TCAR also needs to have a suitable SDLC model to develop the system in the most complete and professional way. There are many models that are suitable for my project like waterfall, v-model, agile. But I decided to choose waterfall model for TCAR.

According to Adetokunbo and Basirat Adenowo, waterfall model is a close model for developing applications. It follows the life cycle with a linear and sequential approach. Provides peace of mind when developing because it divides into specific stages. Specifically, there are 5 stages, requirements and specifications, system design implementation, testing, deployment, maintenance with each stage being started with a waterfall, that is, finishing the above stage and then continuing to the next stage (Adetokunbo A.A. Adenowo, 2013).

### 3.1. Justification of the suitability of a methodology or a framework followed:

After collecting the required data and functions of TCAR, I decided to choose the Waterfall model to develop for the following reasons:

- Firstly, well-defined project development structure with:
  - Requirements and specifications.
  - System design.
  - Implementation.
  - Testing.
  - Deployment.
  - Maintenance.

- Secondly, in this model there is a requirements analysis phase and a specification document to make it easier for me to identify the functional and non-functional requirements that TCAR should have. This keeps me from missing out on important functions and clearly identifies each project function.
- Thirdly, this model helps me in the design phase, it makes it easy for me to build designs like use case diagrams, entity relationship diagrams, wireframe on the front end, and so on. This phase is also the stage of clearly understanding the technologies that the system needs with the right purpose to proceed in my application development phase. During this time is also the time to clearly analyze the risks and ways to overcome the system. With this model, the system will clearly define the completion time of the project after how long with the initial goal set out.
- Fourthly, after clearly analyzing and designing the project requirements, I will only need to focus on developing the application without worrying about other risks that may occur because of the requirements well defined at the system analysis and design stage.
- Fifthly, the waterfall model uses a planned approach in each phase, so the test steps are planned in advance before implementation. This makes it easy for me to test the functions sequentially to see if there are any problems.
- Finally, after testing how is the system running, are the requirements accurate to the one I set out for TCAR? I will deploy application and sum up all the requirements of TCAR again. With this model, the system will clearly define the completion time of the project after how long with the initial goal set out.

# 4. Literature review:

## 4.1. Approach to literature searching:

With research sometimes encounters a lot of difficulties because the copyrights or maybe a certain book is very expensive which makes the research in this report also become very difficult for me. However, the Greenwich university also provided me with a premium account so that I could read the material for free make it easier for me to find data.

To carry out this research, it is absolutely essential that I make sure that all the information that I collect is reviewed from reputable authors or major web outlets or experts in the field academically personally quoted. Research is heavily focused on tourism because I am developing travel car rental application for both domestic and foreign people. What are the current travel problems, covid 19, will the system be able to earn a high profit after TCAR is born? Furthermore, I also look for books or websites that show different charts and graphs on tourist arrivals and car rental needs of customers in Vietnam with intuitive and relevant discovery. All the information collected in this report was searched by me in google scholar, journals, official websites for academic reports, and so on.

## 4.2. Technology used for TCAR:

With TCAR will proceed to develop Web applications according to the trend of demand from users. In this section, I will review and review which technologies are suitable for the project.

### 4.2.1. Web app:

According to Kambil, for software development, web app technology seems indispensable for every application for users. With the development of technology like a storm, now practical applications have developed from 1.0 to web 5.0 (Kambil, 2008).

About web 1.0:

- Web 1.0 for the most part is a person who just writes content, and it is provided by the server's file system and is just a static web page run by ISP or hosted with free web services.
- Tables and frames are used to align content and position on a page.

About web 2.0:

- Users can retrieve and categorize the information for free.
- Dynamic content with information transferred between the website owner and another user of the website.
- The APIS are developed for self-use.
- Social networks are born with sharing and thoughts from one user to another.

About web 3.0:

- Semantic Web with the need to create and connect apps through search and analysis according to language instead of having to use specific keywords or numbers.
- Artificial intelligence and 3D graphics are formed. Thanks to that, games and ecommerce websites flourished.
- In particular, it can be accessed by many applications and used anywhere.

About web 4.0:

- Web 4.0 is considered an open, interconnected, or intelligent web and it is mobile web.
- It interacts with users as people communicate with each other.
- The entire web is an single operating system and this is where all information is transferred from one point to another.

About web 5.0:

- Web 5.0 is especially emotional because it will remove all barriers, helping us to not have much distance between a website and people.

- Gather stimulation and the ability to receive information from users more quickly and freely.
- Web 5.0 will become a manipulative and potentially disruptive space for individuals.

#### 4.2.2. Language JavaScript:

Programming language is a basic thing that every programmer will know and understand well to develop an application in the field of information technology. There are many types like C#, Java, JavaScript, and so on. But in this article, I only mention the JavaScript language. According to MDN, JavaScript was developed in on March 31, 1685 (MDN, 2022). It is a dynamic programming language used for many purposes such as web development, in web applications, for game development, AI, IOT, and so on. In particular, there are many frameworks using the JavaScript language, which are ReactJS, VueJS, Agular on the front end and the highlight of the back end is NodeJS.

#### 4.2.3. Front-end (ReactJS):

For the front-end, in addition to the basic knowledge of HTML and CSS, it is impossible without JavaScript language. But it is not enough to choose JavaScript, because this language has too many frameworks that are not inferior to each other such as ReactJS, Angular, Vue or Svelte. Each framework will have its own advantages and disadvantages. In the scope of the project, I just mentioned about ReactJS. According to Staticta, up to now, ReactJS was developed in on May 29, 2013, and currently has version 18.0.0 and ranks first in the frameworks far ahead of JQuery ranked 2nd (Staticta, 2022),.

According to Peerbit, there are notable strengths of ReactJS (Peerbit, 2022):

- Programmers can code individual parts and all the changes made will not be able to cause the logic application.
- React code is more maintainable and flexible thanks to the modular structure.
- The ReactJS code is reusable, which makes it quick and uncluttered to write code.
- ReactJS is also heavily supported by 3$^{rd}$ parties and is supported by many CSS libraries such as bootrap, taiwindCSS, ant design, and so on.
- Handling dependencies.
- Great cross-platform support.
- Offers great tools for developers.
- Design focused on user interface.

#### 4.2.4. Back-end (NodeJS):

When it comes to JavaScript and ReactJS languages, it is impossible not to mention NodeJS in this report. NodeJS was developed on May 27, 2009, as of now it is using Node.js version 16.13.1. NodeJS is the foundation of chrome based on the JavaScript language. It runs cross-

platform on Microsoft Windows or Linux. It also provides rich libraries of different JavaScript modules just like ReactJS, which greatly simplifies development for web applications on the back-end side (TutorialSpoint, 2022).

There are notable strengths of NodeJS:

- The API uses asynchronous, and event driven. Just to be clear, a NodeJS server will never wait for the API to return. The server that responds from the previous API call will return the next go-to server after calling it.
- No data buffering is ever done, and the application simply outputs the data in pieces.
- NodeJS uses an event loop with a single-threaded model. It can handle a much larger API than traditional servers like Apache HTTP Server.
- NodeJS be able to open, read, create, write, delete, and close files while they are on the server.
- Perform query, edit, delete, add data in basic management systems such as: Microsoft SQL Server, MySQL, MongoDB, PostgreSQL, and so on.

#### 4.2.5. Express:

According to Lab, express is a web application framework combined with NodeJS. It is built on top of Node.js integration module HTTP to set up routes and handle the request or response cycle of the API returned from the front-end (Labs, 2022)**.**

There are notable strengths of express:

- Listen for requests from client to server.
- Responds to the type of HTTP verbs that a certain function does.
- Route the paths of a web page with the combination of HTTP method and URL pattern associated with request.
- Express also uses middleware in order to extend its abilities with self-written code.

#### 4.2.6. Database (MongoDB):

For the database, there will be two types of SQL (Structured Query Language) and noSQL (Non-Relationship Structured Query Language). But in this report, I am only talking about MongoDB a kind of no SQL database. According to Wikipedia, MongoDB was developed on February 2009, as of now it is using version 5.0 (Wikipedia, 2022). MongoDB is an open-source database created as a model that is more advanced in speed, features, and so on than the RDBMS. Remarkable, NoSQL has JSON data type, and this is a key and value data type with scalability and fast performance that is unbound by foreign key generation, primary key like SQL type should be favored and used very commonly.

There are notable strengths of MongoDB:

- Query.
- Replication.
- Load balancing.
- File Storage.
- Rally.
- Server-side JavaScript execution.
- Collection size limit.
- Transaction.

#### 4.2.7. Conclusion about technology:

After doing some research, I feel like the MERN toolkit (Mongo DB, Express, ReactJS, NodeJS) stack will be what I develop in my project. Because this toolkit is too much of a necessity and is completely supportive enough for the TCAR application. My entire application just built with JavaScript language alone was able to build both machine side and server side. MERN is also strongly supported by 3rd parties that come with the necessary libraries to support the development of a web application. Because the time in my project is only 3 months, using this toolkit can completely meet the goals set by my project and can complete the project on schedule. In addition, web 3.0 technology will be applied by me to develop the system because it completely meets the requirements that TCAR needs, can exchange information from the data of many different servers with each other stable performance.

## 4.3. Market analysis:

With the analysis of the tourism issue clarified in section 1.1. I will clarify more about the problems, the needs and the successes and remedies in the field of car rental in Vietnam.

#### 4.3.1. Success has been achieved in the field of tourist car rental:

According to Ken, in 2017, the number of cars that have been and are being rented in Vietnam is 12,467. Notably, the number of car rental transactions between a consumer and a car rental company is 1.11 million dong. Moreover, the number of rental cars continuously increased at a rate of 5.39% CAGR and the profit transaction rate from rental service was 4.09% CAGR (Research, 2018).

According to Phuong Thu, in 2018, the cost of renting a tourist car in Vietnam doubled because of the problem of overloading the number of car rental passengers. The demand for car rental has increased rapidly in recent years from 2014 to 2017 across Vietnam. Regardless of vehicle types, from affordable cars to high-end cars, they are rented at extremely high prices, and it is worth noting that the tourist car rental industry in Vietnam is expected to exceed a huge number of in this car rental field. According to Ken Research, Vietnam is a place with great potential for profit for investors using technology in car rental services. Why this field is so hot

is that cities in Vietnam are constantly changing leading to an increase in tourists along with promotions, discounts to attract customers, and this is the key to success (Thu, 2018).

According to Enterprise Holding official web, thanks to the driving factors, in 2018, the US company Enterprise Holdings, the world's largest rental car supplier, announced for the first time that Enterprise Rent-A-Car had opened its business in Vietnam for the first time (Holdings, 2018). Not only successful in attracting tourists from abroad, but Vietnamese people also attract large foreign firms to invest and develop a large car rental company in Vietnam. Enterprise Rent-A-Car follows the point of long-term rental with a private driver. However, currently, Enterprise Rent-A-Car also only has a single head office in Ho Chi Minh City, so renting a car from many cities that customers want has many limitations.

**4.3.2. Current issues and solve issues with the car rental industry:**

Firstly, the emergence of COVID-19 has limited the constantly changing policies applied to global travel and has impacted car rental demand and reduced arrivals for the whole year of 2020 and 2021 in Vietnam (research, 2021), In 2020, the statistics show that our country's tourist car rental industry has experienced a serious decline. We have lost about 80% of foreign visitors and 50% of domestic customers. However, the good news has also come, Vietnam will officially reopen the gates of all tourism fields when it has had 3 doses of covid vaccine for each citizen on March 15, 2022. This is considered a solution and the car rental service will return to normal.

Secondly, with many companies already financially exhausted and unable to get through the tough times, this will again lead to the same overcrowded car rental situation as in 2014-2017.

Third, ensuring the safety of each trip is still a limitation for car rental companies.

**4.3.3. Conclusion about market analysis:**

After a clear analysis of the successes and risks that the project may face, should the project continue to develop the car rental project? I decided to go ahead with this project because the tourism industry was back with plans for the future that needed to be revived. According to Vietnam photo newspaper, the golden opportunity returns for the recovery of tourism development in the period of the new normal (Nam, 2022). Vietnam's tourism industry is looking forward to and confidently welcoming tourists, which means that car rental applications will be really needed to welcome tourists from home and abroad. In this project, I will develop a large-scale tourist car rental application, the rental cars will have many types and are distributed across many different cities in Vietnam for the purpose of bringing profit. For companies that purchase the TCAR app, the system enhances the user experience and puts safety first for every ride.

# 5. Legal, social, ethical, and professional issues and considerations:

## 5.1. Legal:

The purpose of developing TCAR is to avoid overcrowding of passenger cars, make payment easier and avoid the risk of accidents leading to loss of life for tourists. Basically, the project will not face many legal difficulties. The authorities in the tourism and software sectors have agreed to develop the application based on the contractual agreement between the parties. The project will be implemented in the forms determined by the competent authority. Software developers and people involved in the tourism industry join and analyze to develop the application because this is a project based on the foundation of the growing tourism industry in Vietnam. Users' personal information and online data will be safe and secure. I also promise we do not use the data for malicious purposes.

## 5.2. Social:

TCAR application helps support the country's tourism industry to develop further and find solutions to overcome unexpected accident situations for customers. Bring satisfaction to both domestic and foreign customers. Paying money also becomes easy through banking from 3rd party's services and security to avoid unnecessary risks. The application will be used in many cities of Vietnam.

## 5.3. Ethical and professional:

Ethical issues will take precedence, I will agree to a contract to avoid issues such as piracy, patents, trade secrets, copyright, harmful actions, and liability. If application encounters any problem, the user can completely complain and refuse to use the application for any reason. Moreover, TCAR will also meet the information security requests for each user's account and authorized roles in the system. The authorized roles in the system will not be allowed to know about the user's payment card information for the purpose of avoiding the risks that they may abuse the privacy rights of customers for their own gain.

## 5.4. Considerations:

Data storage requirements can increase, and data needs to be managed in new approaches. Data storage systems need to keep up with the pace of development to ensure safe data for users. Furthermore, backups need to be stored separately from the main server so that they are not compromised in the event of a user data breach. I also needed to correctly identify and categorize the content of the TCAR application to make it easier to evaluate the technology. TCAR needs risk assessment and protection because data theft can cause serious harm to a business. The system needs user access control to significantly improve security. If the system doesn't have unnecessary access privileges, they can't do much damage if they are breached.

# 6. Requirements:

Requirements give the project a clear understanding of the required tasks that a software program develops. Determining the requirements will ensure that the project always follows the trajectory set by the goals with the aim of meeting the needs of the customer from the project. Analyzing project requirements helps me understand the problems that existing applications have problems and then propose solutions so that TCAR can meet today's market needs. The key to success is understanding the customer's problems. I needed to learn and define the functional and non-organizational requirements to complete the project with the original set goals and timelines.

## 6.1. Analysis of requirements:

For any part project, it is essential to define and analyze the requirements of the project's proposed system. In this part, I need to analyze all the high-level requirements before proceeding with the project implementation. Here are the project's high-level requirements:

| Requirement ID | High-level requirement | Justification |
|---|---|---|
| 1 | User can register account, login with multiple methods, logout. | All applications must have a registration function, so that the system can confirm who is renting the car. The development of login functionality helps to clearly manage user roles in the system. Based on each role user's permission in the application, they will do different things. Signing in with many ways such as Facebook, Google, GitHub, and so on gives users more options without having to sign up for an account. The logout function helps users protect their personal accounts. |
| 2 | Forgot, reset password. | This function helps users to restore their accounts after a long time when users do not access the application. |
| 3 | See and change profile, change password user. | Users can see and update their profile, even, the password change function helps users to protect their own account when there is any problem with their account. |
| 4 | Manager account user (create, change Password, delete). | The management of user accounts, helping administrators and employees |

| | | |
|---|---|---|
| | | can manage the accounts of drivers, and users. This helps administrators and staff assess whether the application has attracted a number of customers who have engaged in the application or not. |
| 5 | Edit role and delete account user. | The fact that the administrator can change permissions and delete accounts, helps the administrator to update the roles of the staff each time they are promoted and delete the accounts where the staff have retired. |
| 6 | Account statistics, the number of cars booked, and the monthly rental amount. | This helps administrators and staff to continuously update the number of customers who have rented the car and the amount of money collected after each month. It helps the authorized roles in the application capture the revenue for the company after each month. Moreover, employees can easily control the cars rented by each city. |
| 7 | Manager car and assign car to driver. | Roles with permissions in the application can add, edit, delete cars according to the driver's free days. Drivers will be assigned to their vehicles according to the regions they are working in in that city. |
| 8 | User can find car based on price, seat category type, date, and time available. | The system will suggest the right vehicle for the user for the place that the user wants to rent. Users can search for cars by price, date, and time they want to rent, or vehicles with high ratings. This helps users have a better experience instead of having to read each vehicle in detail to find a car. |
| 9 | Add and remove car from favorites car. | Users can add or remove the cars they will put favorites car in the app. This helps users just come the cars they love |

| | | |
|---|---|---|
| | | instead of having to find the car they were looking for yesterday. |
| 10 | Users book a car and payment online. | The development of this application makes it possible for users to search for cars and pay online with their bank credit cards instead of having to pay directly at TCAR. |
| 11 | See booking. | Staff, drivers, and users can view details of car rental date, car rental amount. |
| 12 | User can read, rating, review car and driver. | The fact that users can read and evaluate will help administrators and departments have the right to understand the negative and positive issues after each customer's booking. This helps to improve service quality for TCAR. |
| 13 | Driver sees my car assign. | For driver to see the car they are in charge of. |
| 14 | Chat message with authorized roles in the system. | This request helps TCAR members to communicate directly on the application. They can create a group to solve the problems that TCAR is facing instead of having to use other applications such as Facebook, Zalo, and so on. |
| 15 | Chatbot auto for user | Support users with troubles in car rental |

Table 1: Analysis of requirements.

## 6.2. Existing solutions:

After a general analysis of the requirements in section 5.1 can be developed for TCAR. I want to research about some applications that exist in the market today, whether it is enough to meet user needs, strengths, and weaknesses of these applications. And from there I will clearly define what TCAR needs to get the project on the right track.

### 6.2.1. Rentalcars.com web:

According to Rentalcars.com web, Rentalcars.com app was developed in 2004 as TravelJigsaw. In just four years, they were making 1000 bookings a day. Currently the system is based in Manchester. Fast-forward to this year (2022) and they will make 8 million bookings a year (and

counting) at more than 60,000 locations large and small in 160 countries (Rentalcars.com, 2022).

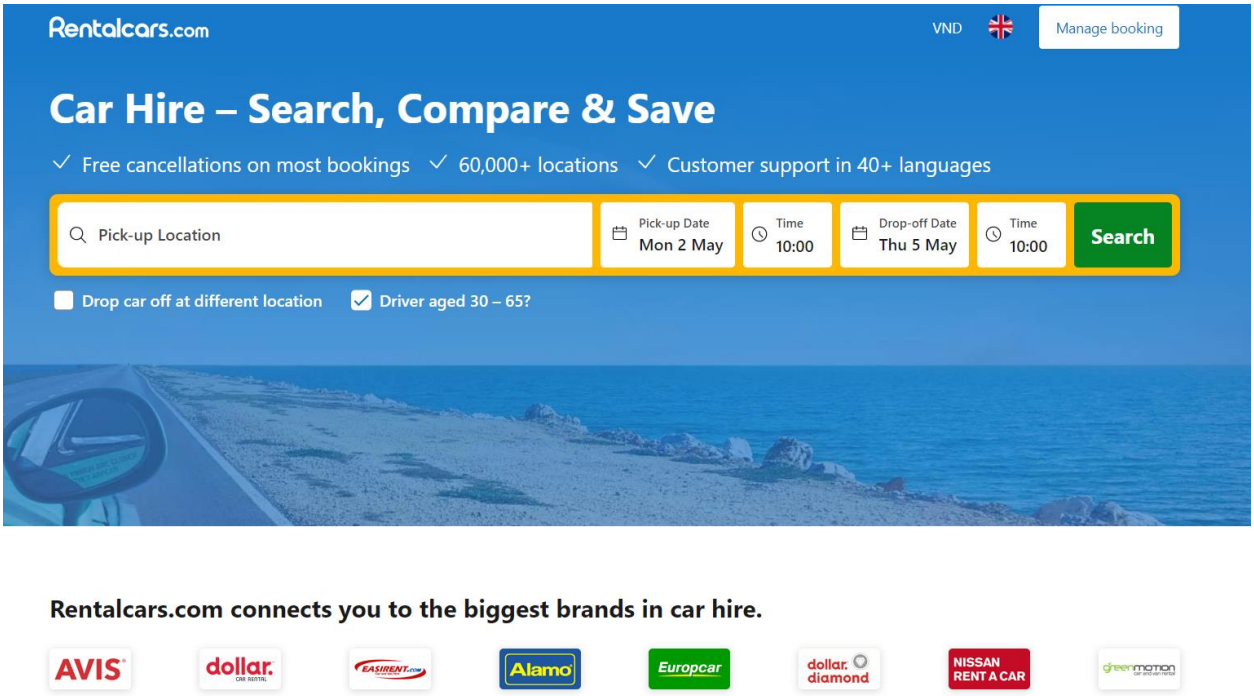



Figure 1. Rentalcars.com web.

Pros:

- Has all the necessary functions such as registration, login, payment, and search car.
- The user interface looks intuitive with well-coordinated colors.
- Large scale car rental with cars rented across many countries.
- Support online payment with many types of credit cards

Cons:

- The detailed instructions for renting a car are too difficult and there are no clear step-by-step instructions.
- I have tried creating an account and this the account creation function don't know if it works or not because I don't see any response from the app.
- Users are required to create an account before they can log in, because they can log in with facebook or google.
- The rental price for each trip is too high.
- The app does not have a messaging function yet.

### 6.2.2. Turo web:

According to Turo, Turo is a major touring car rental app that is the world's largest marketplace in the United States, Canada and the United Kingdom. Turo has 7,500+ rental cities across the US, Canada and the UK. Turo's annual salary is up to $ 10K/year for trips (Turo, 2022).

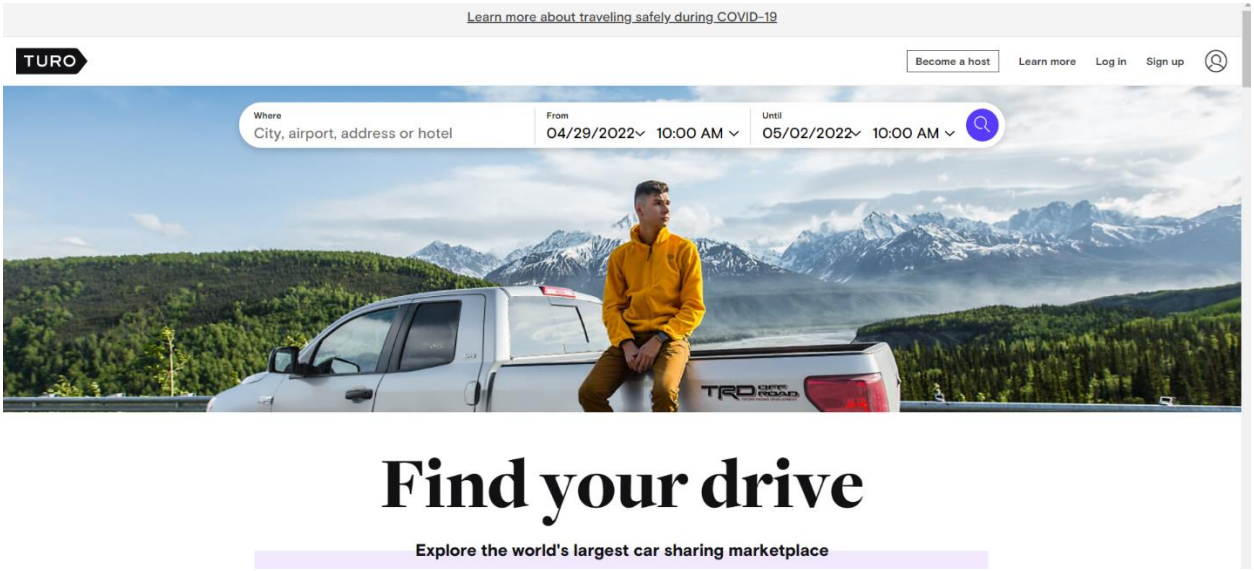



Figure 2: Turo web.

Pros:

- Has all the necessary functions such as registration, login, payment, and search car.
- Beautiful interface with animation images.
- Fast and secure car rental system through user payment codes.
- There is a map showing the cars in the slot in each area.

Cons:

- Finding a car is still difficult because the data for each area is too slow.
- There are too many ads that make it difficult for users to confirm if this is a car rental app or not.
- There is no chat function for the purpose of discussing permissions in the application.

### 6.2.3. Chungxe web:

Chungxe is one of the first companies and startups in the field of car rental in Vietnam. Car rental sales for each year are about VND 4k/ year for trips. And currently renting tourist cars all over the cities in Vietnam.

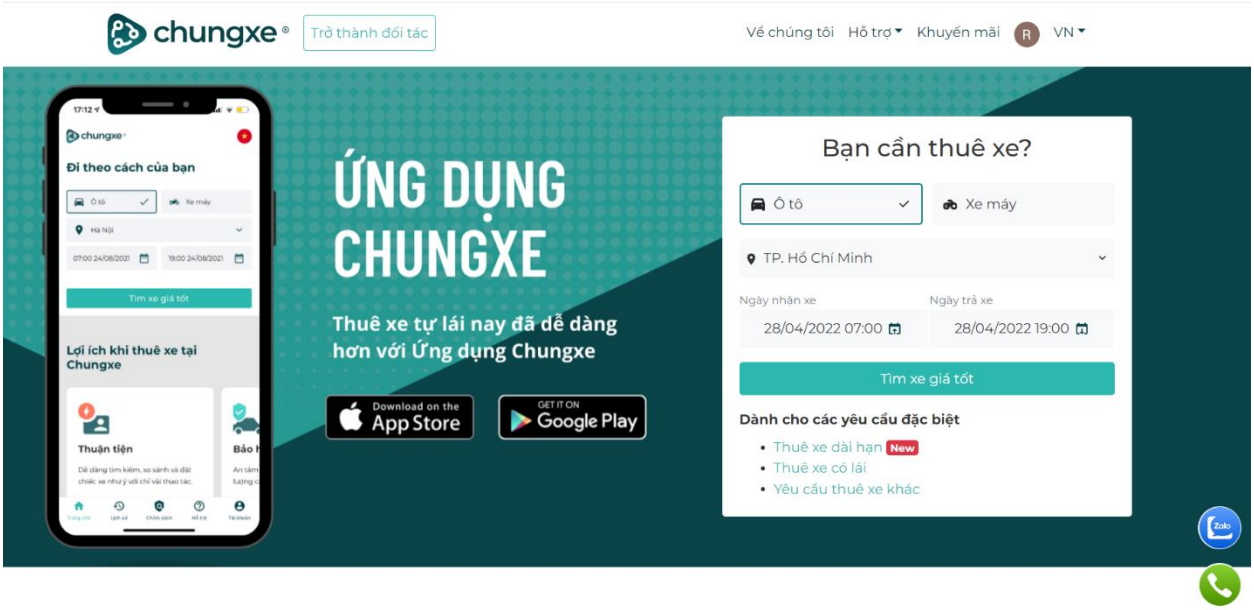



Figure 3: Chungxe web.

Pros:

- Has all the necessary functions such as registration, login, payment, and search car.
- Can login by 2 methods google and facebook.
- Application support on both web and mobile.

Cons:

- The interface and experience are sketchy, it seems that this application prioritizes server logic more.
- There is no chatbot feature to assist users in finding cars.
- The payment feature is not supported for many cards owned by foreign cities.
- The forgot password function has only one method, which is to receive from email instead of supporting users to change the password with their real phone number.

**6.2.4. Conclusion:**

From the requirements and existing products on the market, I decided to improve and overcome the elements that a car rental application need. In terms of revenue to be for TCAR, it is a development step with TCAR application version two because the version one application is only for bringing products to the market to reach users. Of course, TCAR version one will not be able to compete in sales with apps that already exist in the market.

Across the products I researched, what really mattered to me was to choose the right features and functionality for TCAR based on the advantages and disadvantages that existing apps have. Experience will be TCAR's top priority, so TCAR will optimize functions such as finding a car,

paying in the most specific and easy way for users. With the disadvantages of finding a car difficult from current applications, I decided to assist users with an intelligent function that is a chatbot, which helps users to solve all questions and concerns when looking for a car. The content of the interface will only focus on the cars without wandering or promoting unrelated issues like the Turo app. The system will have functions with 2nd methods such as login with Google or Facebook, get forgotten password code from email or SMS, in addition can pay with many types of cards through 2 service portals Stripe and Braintree. Furthermore, admins and employees can manage the car, bookings, and message real-time to discuss the goals the company is striving for to surpass existing applications. The car booking will automatically update the status based on the date that the user rents the car. In addition, the car rental price will be lower than the common ground of the current car rental application for the purpose of attracting customers.

# 7. Business requirements:

## 7.1. Overall picture:

An overall picture can show all the actors of a TCAR application. It is used to identify issues and requirements for TCAR. An overview of these processes will make it easier for me to develop the system as originally intended. Thought bubbles explain the triggers and how to overcome them. The square dialog contains all information about their responsibilities or purposes when using system. Firstly, immigration office is included in the picture because the immigration office makes sure that system will not violate any data protection laws or any other laws. Secondly, business analyze will be responsible for analyzing the tourism situation in Vietnam based on the required analysis data. The analysis will be based on the Covid-19 situation. Furthermore, business analyze will also look at the application problems of existing businesses, thereby offering suitable solutions. Thirdly, admin, staff, data analyze, driver will give required opinions on suitable functions for the project such as clear statistics on the amount of users' money after each month, and so on. Fourth, user will be the one to experience the application and give feedback to the system for TCAR to improve the system. If the system attracts customers, the revenue will go to TCAR company to improve system. Finally, TCAR company wants the system to bring in large revenue for the company, requiring the system to have a user-friendly and easy UI & UI and the system to be used online with booking management functions, pay online and so on.

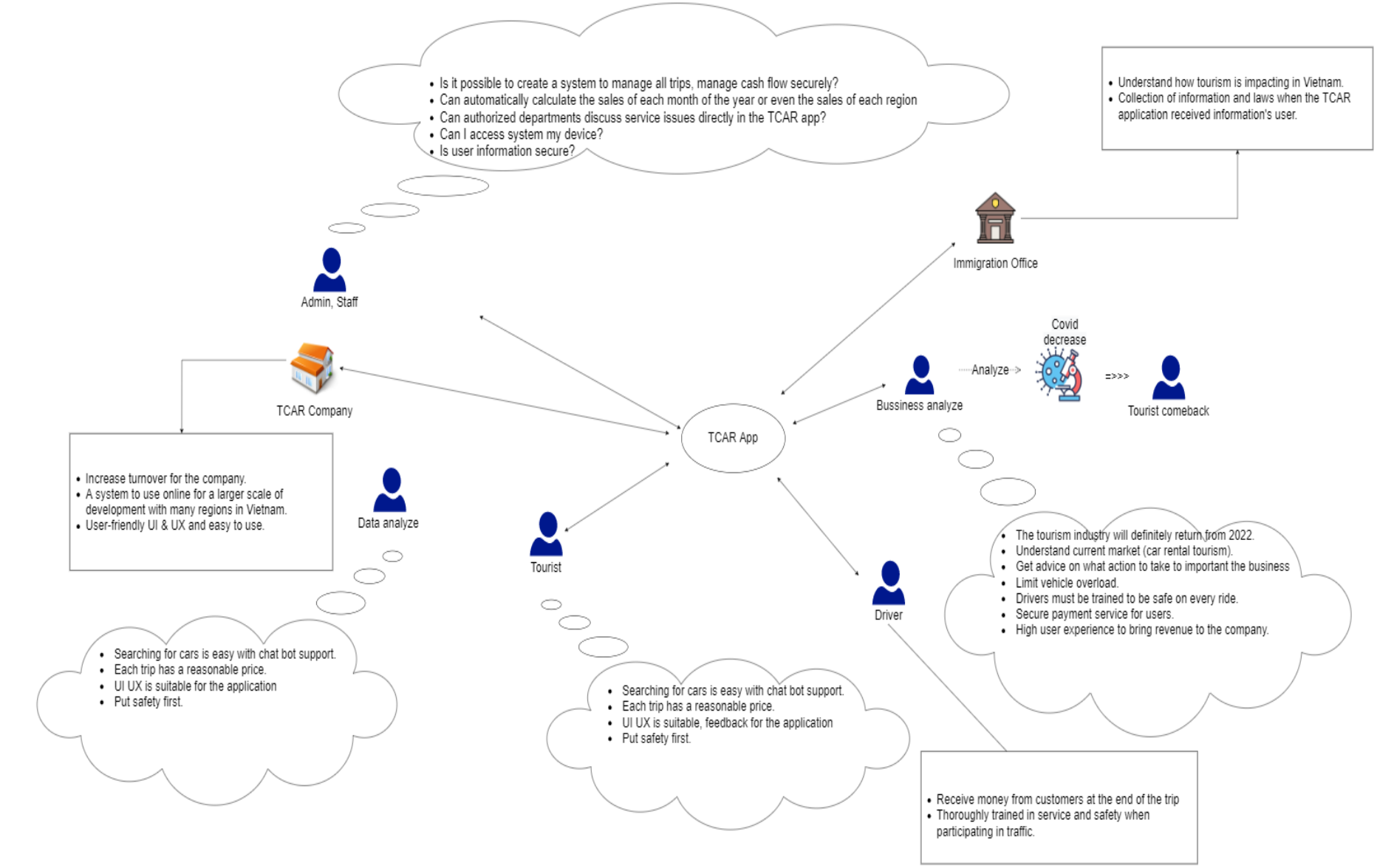



Figure 4: Overall picture of business requirements.

## 7.2. Functional requirements:

Below are all functional requirements for TCAR:

| No | Functional requirements | Justification |
|---|---|---|
| 1 | Register. | This function helps the system to easily manage and store user account information. Managing user accounts will help the system know who is renting the car. |
| 2 | Login normal and gg or Facebook. | The login function helps the system to easily identify the user's role in the system. Logging in with google or facebook will give users more quick login methods instead of having to enter their account or password. This function will help users to book a car, rent a car and payment. |
| 3 | Logout. | Logout out will help users protect their accounts after each login to the browser. |
| 4 | Forgot (email or phone) and reset password. | For accounts that have registered and logged into the application, however, there are some people who do not access the application for a long time, they may forget their password. Therefore, the function of forgetting and recovering passwords for users is necessary. |
| 5 | See and change profile, change password account. | Users can see and update their profile, even, the password change function helps users to protect their own account when there is any problem with their account. Therefore, this function is necessary. |
| 6 | Manager account staff or driver (create, change password, delete). | For TCAR app, account management of the roles is extremely important. The higher-level roles will be responsible for managing the accounts of the lower roles. |
| 7 | Edit role and delete account user. | This function is necessary because the admin will have the right to change the permissions of the members in the system. Because employees will be promoted after 6 months, it is necessary to change user roles. |

| | | |
|---|---|---|
| 8 | Account statistics, the number of cars booked, and the monthly rental amount. | It is really necessary to make a list of the total number of accounts, the number of tenants and the amount in each month, because this helps those with rights in the system to determine the company's revenue source after each month, from which to develop new strategies in the future. |
| 9 | Manager car and assign car to driver. | Car management makes it easy for administrators to see vehicle status, which cars are operating where, and which cars are being rented. Each driver will only own one vehicle with the city they work in. |
| 10 | User can find car based on price, seat category type, ratings, date, and time available. | This function is needed so that the user can search for a car within the time period that they want to book a car. Searching by car price and type, or cars with high ratings, helps users have more options when searching for cars. |
| 11 | Add and remove car from favorites car. | This function makes it possible for users to add or remove cars that they will put their favorite car in the application. This helps users only need to go to the favorite cars cart they love instead of having to find all the cars with free time that they selected yesterday. |
| 12 | Users book a car and payment online. | For TCAR application, car rental and online payment are mandatory. Online payment helps the system determine which users will definitely rent that car and users can more conveniently pay online instead of users having to go directly to TCAR to payment. |
| 13 | User can read, rating, review car and driver. | This function is necessary because it will reflect the condition of the vehicles after each trip such as how the driver is, whether the condition of the vehicle is stable and so on. From there, offer ways to solve problems for users to have a better experience for each ride. |
| 14 | See booking. | Roles such as staff, driver and user can see the details of the rental date, the amount of the |

| | | |
|---|---|---|
| | | rental, who rents the car, and which driver will take care of that car. |
| 15 | Driver sees my car assign. | For driver to see the car they are in charge of. |
| 16 | Chat message with authorized roles in the system. | This request helps TCAR members to communicate directly on the application. They can create a group to solve the problems that TCAR is facing instead of having to use other applications such as Facebook, Zalo, and so on. |
| 17 | Chatbot auto for user. | This function is required to overcome the user's car search that the existing applications in the market do not have. In addition, the chatbot can also answer users' questions about TCAR. |

Table 2: All functional requirements for TCAR.

## 7.3. Non-functional requirements:

Below are the non-functional requirements for TCAR:

| No | Type | Non-functional requirements | Justification |
|---|---|---|---|
| 1 | **Security** | The user's password must be hashed with Bcrypt or Crypts. | The system needs to hash the password to secure the user account in the safest way. |
| | | User forgot password, the link to create a new password must be sent to that person's email. | For the purposes of user safety, if system do not send the user's own email to change the password, others will be able to abuse and take the account of the person who lost the account. |
| | | All functions of receiving OTP codes from SMS and Email are only valid for 30 minutes, logged in users will only have 1 day time. | The sections that receive codes to register, receive SMS or login status are stored only for 1 day to protect the system's rights to use the service. |
| | | The system must not store the user's credit card information when the user makes an online payment on the TCAR application. | The card information after payment will not be stored in the application to avoid the staff in the system to abuse and use the |

| | | | |
|---|---|---|---|
| | | | customers' personal payment accounts. |
| 2 | **Usability** | The buttons, icons, and content for each action when the user interacts with the system will be carefully selected and analyzed so that the user can easily identify the features that the application wants to convey. | Buttons with icons are clearly essential to help the user understand what actions the application wants for the customer to interact with the system. |
| 3 | **Error tolerance** | Validate the data type on each field where the user enters data in the form and immediately alert the user if the user enters incorrect input data such as registration, login, forgot password, create car, and so on. | This helps the data stored in the database match each value that the database needs. |
| 4 | **Availability** | Guaranteed to operate 24/7 and not depend on the downtime of any 3rd party products. | The 3rd parties of the system such as google, facebook, email and payment gateways will be linked by me, and these 3rd parties will support to ensure 24/7 operation so that users do not have problems when using TCAR. |
| 5 | **External interface** | All data must be integrated via API and API must return results in JSON. | This helps TCAR's client-server development model to unify the data that the front-end will transmit data to the back-end, and the back-end to receive data from the front-end. |
| 6 | **UI & UX** | The system needs to be easy to understand and user-friendly. | It is certain that all end users will have problems using the system on the first try so the system needs to prioritize ease of use. Ease of use can ensure that system is easily accessible to the end user and interacts with the application faster. |
| 7 | **Legal** | The system must be required to limit abuse problems and comply with data | Ensuring data must delivered correctly and this requires the |

| | | | |
|---|---|---|---|
| | | protection laws when system processes data. | system to follow the appropriate laws and legal guidelines. |
| 8 | **Performance** | For data entry form: Only maximum to 25 data fields, must be interaction with external systems, no should too complicated data calculations, maybe store data directly into database and no extra storage more than 10MB images. | The data fields should only be in the range of 20 or less because 20 or more fields will cause user's trouble. In addition, external interactions will be denied so as not to affect the data of the system. And the images will not be more than 10MB to avoid data limit in the database. |
| | | For the output screen: The front-end cannot directly query to the database but must go through the API. The system should limit requests query that are too complex or queries from any external system that is not belong about in the TCAR system. Moreover, the data can only be displayed up to 15 rows, each row is only 8 columns, and each data is only under 120 characters in length. | The data should only display up to 15 rows and 8 columns with the sole purpose of making the user interface more beautiful and clearer with each piece of data. |

Table 3: Non-functional requirements for TCAR.

# 8. Analysis and design:

## 8.1. Architecture:

The architecture that TCAR will use to develop the system is the client-server. The reason TCAR uses the client-server architecture:

- Hardware has nothing to do with software, apart from the only requirement that the server has a higher configuration than the clients.
- Support for the system with a variety of services and convenience by remote access.
- Centralized system with all data in a single place and ensure data integrity when something goes wrong.
- Users be able to access remote data, perform simple operations to send and receive files and search for all information.
- Provides a good user interface and management system for organizing files.

- Ensure that the server is replaced, restored, upgraded, and moved without affecting the client because the client-server model is based on the distributed model.

## 8.2. High level design:

### 8.2.1. Assumptions:

Project TCAR, it is necessary to identify the conditions that are most likely to occur as the life of the project occurs. It is also considered a factor in the planning phase. Here are my assumptions and workarounds for TCAR.

| Assumptions | Manage assumptions |
|---|---|
| I could be positive for Covid-19, which is bad. This probably violates the assumption that all my project can be delayed for 2 weeks. | Conduct a hypothetical analysis for risk management planning to come up with a contingency plan if the actual plan doesn't work. |
| The project's estimated time can be met, and the project will be completed within the scheduled time. | Always track and analyze the progress of the project to track the progress of the work. Use project management tool (Jira) to visualize and map out the project schedule to identify dependencies to derive project constraints and assumptions. |
| The quality of equipment, software, and hardware must be in good working order throughout the life of the project. However, some devices, software, hardware may be degraded thereby affecting its operating condition. | Record and track project assumptions in the Project Assumption Log to maintain and monitor the quality of equipment, software, and hardware. From that, analysis of project assumptions is drawn. Plan, predict, and monitor the scope, specifications, and possible changes as the project progresses. |
| The software development process of the project must meet the needs of the customer and the initial goals of the project. | Always refer to the Software Development Life Cycle (SDLC) when designing software projects and analyzing user needs to identify the essentials with customer needs. |
| The project always works with the usual schedule regardless of weather changes. | The project always works with the usual schedule regardless of weather changes. |

Table 4: Assumptions of project.

### 8.2.2. Endpoints:

An endpoint is a vulnerable point that can be used by cybercriminals to infiltrate the system. Therefore, clearly defining the endpoints is essential for TCAR. TCAR will have 4 connection endpoints that are laptops, desktop computers, smartphones, and tablets.

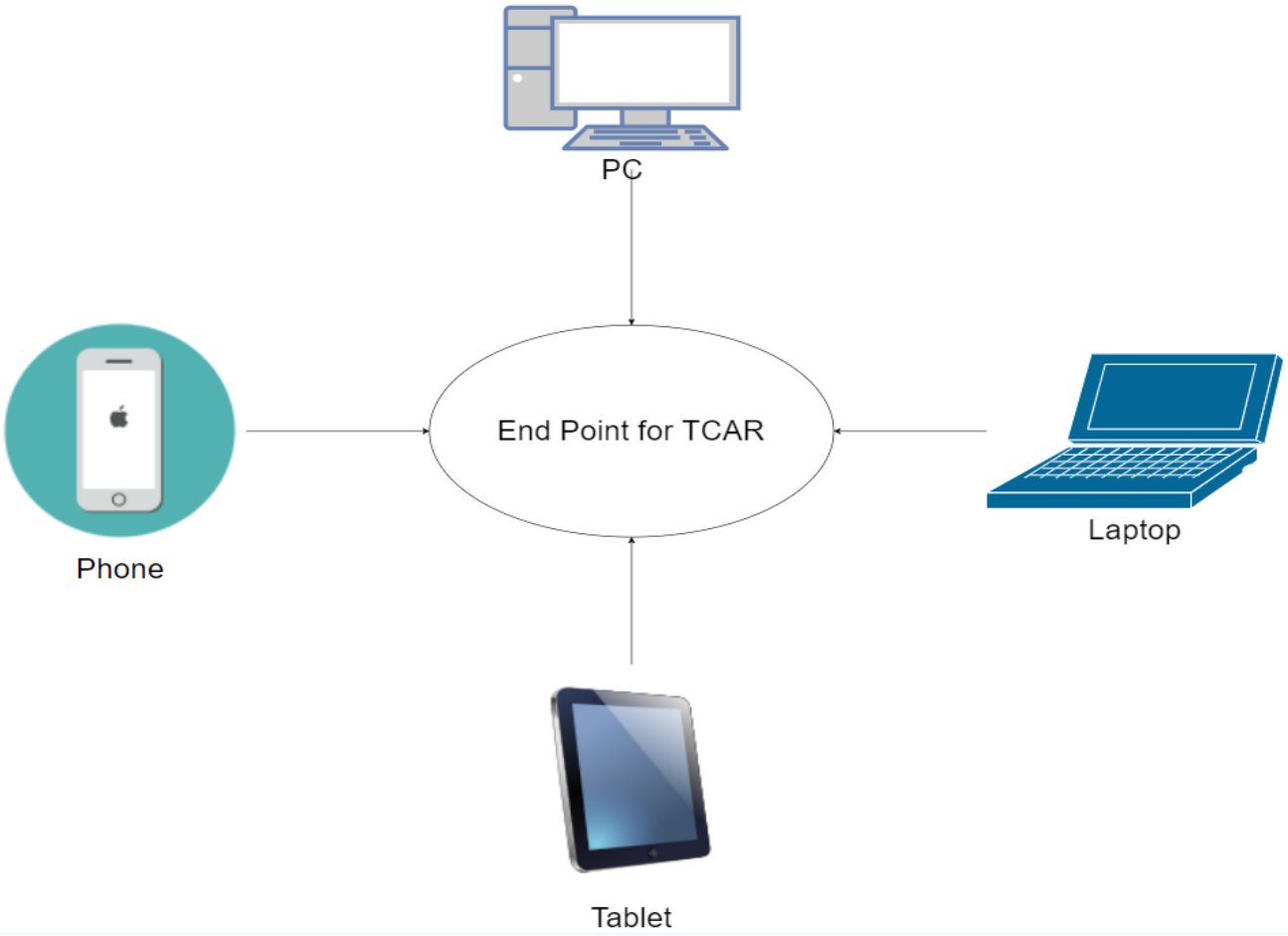



Figure 5: End point for TCAR.

### 8.2.3. 3rd party services:

#### 8.2.3.1. Google cloud platform:

According to TechTarget, Google Cloud Platform is a platform of cloud computing technology that allows individuals or organizations to build, develop, and operate their applications on the software system created by google (TechTarget, 2022).

Google Cloud offers some of the following key products:

- Storage: Cloud SQL, Cloud Storage, Cloud Datastore, and so on.
- Compute: App Engine, Compute Engine, and so on.
- Services: Translate API, Prediction API, and so on.

- Big Data: Cloud Dataproc, BigQuery, Cloud Dataflow, and so on.

The reasons I chose Google cloud platform for the TCAR project:

- Google Cloud Platform helps user easily login with google in TCAR system. Google Cloud Platform is supported by Google's automatic protection that works during sign-in which helps ensure account authentication and keeps users' google accounts safe. Furthermore, it also has two-step verification (2SV), which secures and protects the account from more sophisticated attacks. When 2SV is used, users are required to authenticate in two steps, using something they know, such as a password, and using something they have, such as a code or hardware device.
- Google Cloud Platform provides me with a free Gmail sending service so I can build functionality that requires the user to go to Gmail to authenticate requests from the TCAR app.
- Google Cloud Platform leverages the GCP cloud, which decrease me of the tasks associated with maintaining the physical infrastructure during development.

#### 8.2.3.2. Meta for developer:

Meta for developer is an application to connect with customers effectively using prominent platforms such as Facebook, Instagram, WhatsApp, and so on.

The reasons I chose Meta for developer for the TCAR project:

- It helps user easily login with Facebook in TCAR system. Moreover, it's easy to set up the best Facebook business tools in the platform you must create a comprehensive solution.
- The required login accounts must be real and securely authenticated.
- Strongly supported with libraries of two frameworks ReactJS and NodeJS.

#### 8.2.3.3. Twilio:

Twilio is a user engagement platform used by developers and businesses with various sizes. It supports services such as text, video, chat, voice, and email through API easily (current, 2022).

The reasons I chose Twilio for the TCAR project:

- Support me in OTP authentication with forgot password function when Twilio sends SMS to customers.
- All customer phone numbers requested must be correct for each region and correct.
- For API developers open up flexible communication solutions for businesses of all shapes.

#### 8.2.3.4. Nodemailer:

Nodemailer is a module for Node.js applications whose purpose is to enable easy emailing. It is at the forefront of solving application mailing problems for NodeJS developers (Nodemailer, 2022).

The reasons I chose Twilio for the TCAR project:

- Helped me in easy setup and development of functions such as registration, forgot password, and so on in TCAR app.
- Much focus on security because it uses OAuth2 for authentication.
- The information is supported with plain text and can add attachments to the content that the mail wants to transmit to the user.

#### 8.2.3.5. Braintree:

Braintree is a payment platform that makes it easy for users to payment in today's software applications. Braintree provides a modern service to replace the traditional model of sourcing payment gateways and merchant accounts from different vendors. It makes one-touch payments with the SDK and accepts foreign currency requests (Braintree, 2022).

The reasons I chose Braintree for the TCAR project:

- Supported with React and NodeJS libraries.
- Payments can be accepted across all debit cards or major credit, local payment methods, digital wallets, and 130 currencies across 45 countries deposit with ACH.
- Braintree provides periodic billing reports and integrates with many third-party applications. It ensures PCI compliance and provides secure data security.

#### 8.2.3.6. Stripe:

Stripe is considered the leading secure online payment platform for businesses and secure credit card processing of all sizes. It supports card performance and secure money management (FreshBooks, 2021).

The reasons I chose Stripe for the TCAR project:

- Accept payments without the risk of disruption to cash flow.
- Free to use services from customers and developers.
- Works with all currencies, banks, credit cards and wallets.
- I didn't have to write additional subroutines to integrate Stripe's API in the original codebase because Strip provides client and server libraries from React and NodeJS.

#### 8.2.3.7. Dialogflow:

Dialogflow is a service provided by Google with the aim of helping developers develop applications that communicate between systems and people through specialized conversations such as conversation or text. It uses artificial intelligence (AI) with purpose to help analyze user implications to solve user queries (Cloud, 2022).

The reasons I chose Dialogflow for the TCAR project:

- I will easily develop a chatbot to answer all user asks in TCAR app. A chatbot is considered an intent powered by Dialogflow that represents a mapping between what the user enters, and the action to be taken by the software.
- Support for writing APIs with NodeJS is easy without me having to write too much logic myself.
- Having a chatbot will bring interesting experiences to users when using the TCAR application.

### 8.2.4. Overall picture:

This is the overall picture of the TCAR app. Users access the application to interact with the functions in the system. The user will communicate with the client-side and the client-side will receive the user's interactions returning requests to the server-side. The server side will have Routes, Middleware, Controller, Utils, and Model sections and they interact with each other to query data in the database. In addition, the server-side also interacts with the 3rd Party-Service for each necessary function that the controller requires.

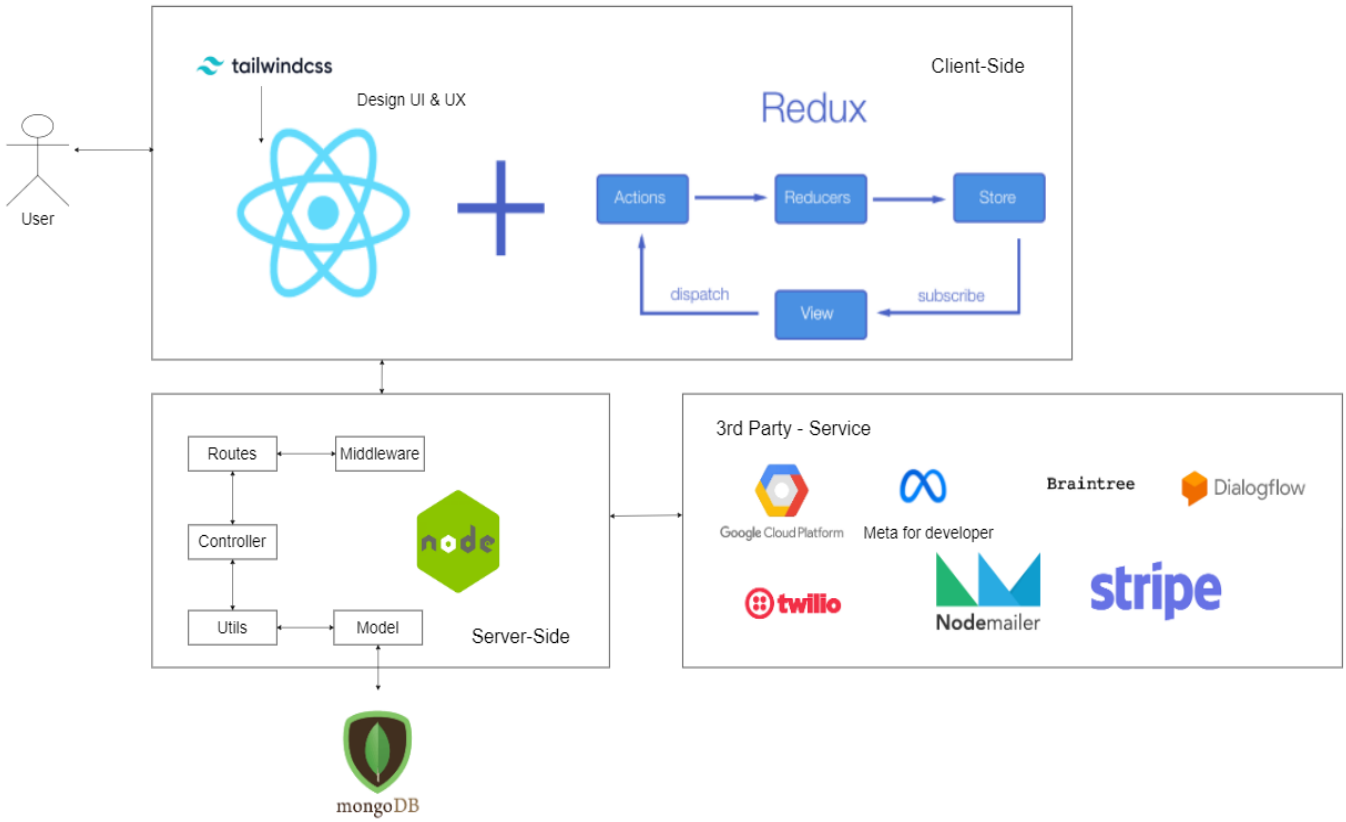



Figure 6: Overall picture.

## 8.3. Technology choices:

### 8.3.1. Front-end:

#### 8.3.1.1. ReactJS:

After clearly researching front-end technologies, I decided to use ReactJS framework by the following reasons:

- I gained some basic knowledge of JavaScript language during my studies at Greenwich university.
- React JS is designed to provide high performance. Core of framework provides virtual DOM programming and server-side rendering which will make TCAR run faster.
- React JS can reuse its components. This saves me a lot of time because I do not have to write a lot of different code for the same features.
- Using ReactJS will help improve page load time and fast page rendering speed.
- ReactJS follows the downward data flow for the purpose of ensuring that parent structure will not be affected by any modification in its substructure.
- There are many libraries and 3rd parties that support for ReactJS.
- Easily develop a MERN Stack application.

#### 8.3.1.2. Redux:

After clearly researching front-end technologies, I decided to use Redux by the following reasons:

- Redux makes the results predictable and the results are tightly structured and easy to maintain.
- Redux allows me to minimize the data flows passed with multiple components with the aim of minimizing breaking of child components.
- I can store the entire state of the application with 'store' in Redux. The state transition from component 1 to component 5 will be easy because the state and is saved and the component can fully use component 4's data by the store.

### 8.3.2. Back-end:

After clearly researching back-end technologies, I decided to use NodeJS - Express framework by the following reasons:

- I gained some basic knowledge of JavaScript language and MVC model during my studies at the University of Greenwich.
- NodeJS uses single-threaded event loop model and provides a non-blocking asynchronous architecture without creating any extra threads using less resources. This increases the responsiveness of the application because the application can handle multiple responses from concurrent users at the same time.
- I can use a lot of libraries, ready-to-use code and other resources from GitHub because NodeJS has a large community and is supported by many libraries. This saves me a lot of time and effort coding.
- NodeJS can integrate several third-party applications and services with Express.JS.
- Manage code and data files easily.
- NodeJS uses built-in API to develop HTTP and DNS and JSON server, this is extremely necessary as I will be using MongoDB for project development, integration with JSON data will make it easy to query data from the documents of NOSQL.
- Easily develop a MERN Stack application.

### 8.3.3. Database:

After clearly researching many types of databases, I decided to use MongoDB by the following reasons:

- With MongoDB no complex joins are required, and it is extremely easy to change the document structure in MongoDB, which will help me connect other documents without any difficulty.

- MongoDB does not need to generate primary and foreign keys. NOSQL databases provide an _id field, which is generated by default for every document, and this field is created for the purpose of acting like a primary key. The _id field is the primary key, and it is a unique value.
- MongoDB uses flexible document schemas, and this gives it an advantage when working with complex data types and dealing with real-time data. The fact that it can use the document schema flexibly is due to the fact that it stores its data in many objects in a uniform way.
- MongoDB stores each record in Binary JSON, so using JavaScript libraries like NodeJS, ReactJS or Express is easy.
- MongoDB performs logging with high speed and real-time analytics in use, which makes it easy for TCAR to develop real-time messaging functionality.
- Easily develop a MERN Stack application.

#### 8.3.4. Web app:

After clearly researching many types of web app, I decided to choose web app 3.0 by the following reasons:

- Web 3.0 supports fast and efficient information search.
- The web surfing experience is enhanced with the aim of increasing the user experience of using a web browser. Therefore, web applications will analyze the user's Internet habits and usage to customize themselves to best suit the device, location, and so on.
- Data will be stored on distributed nodes so users will not need to worry about service interruption due to technical or other reasons.
- Working on the internet becomes easier because the internet is personalized.

#### 8.3.5. Operation system:

After clearly researching many types of operation system, I decided to choose Window 10 OS by the following reasons:

- Windows has anti-virus and security features and keeps their system always up to date to be more secure, which keeps my code-developing apps, my code files from losing data, and keeps them secure safe way.
- Windows 10 supports a lot of programming languages. Moreover, it strongly supports the JavaScript language in the TCAR project.
- All the tools for coding can be used well on Windows 10 such as Visual Studio, MongoDB compass, postman, Windows Terminal and so on.

### 8.3.6. Hosting:

After clearly researching many types of hosting such as Amazon Web Services, Google Cloud, Azure, I decided to choose Heroku by the following reasons:

- I can use Heroku for free with tons of addons that are extremely helpful. It provides database, strong support in linking with GitHub in a simple way. Furthermore, it is easy to deploy to the MERN Stack toolkit.
- Great plugin support for third-party apps.
- Moreover, I just need to focus on product development instead of focusing on server and infrastructure management. Because Heroku has built-in support for managing servers and infrastructures.
- Deployment application is always running 24/7 with high performance and site speed.

### 8.3.7. Conclusion:

This is overall picture technology choices:

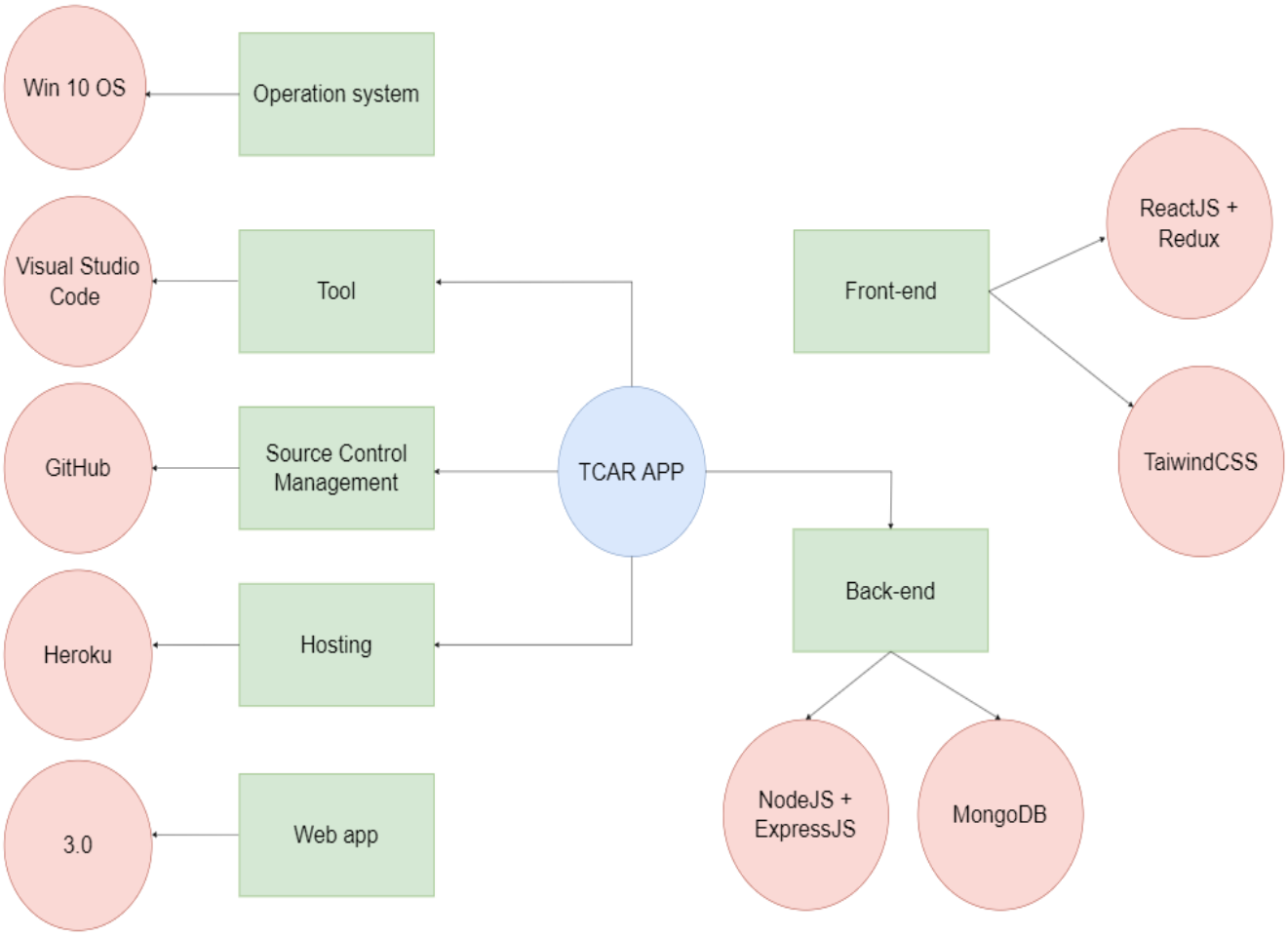



Figure 7: Overall picture technology choices.

## 8.4. Use case diagram:

### 8.4.1. Use case diagram, primary and secondary use-case scenario for admin:

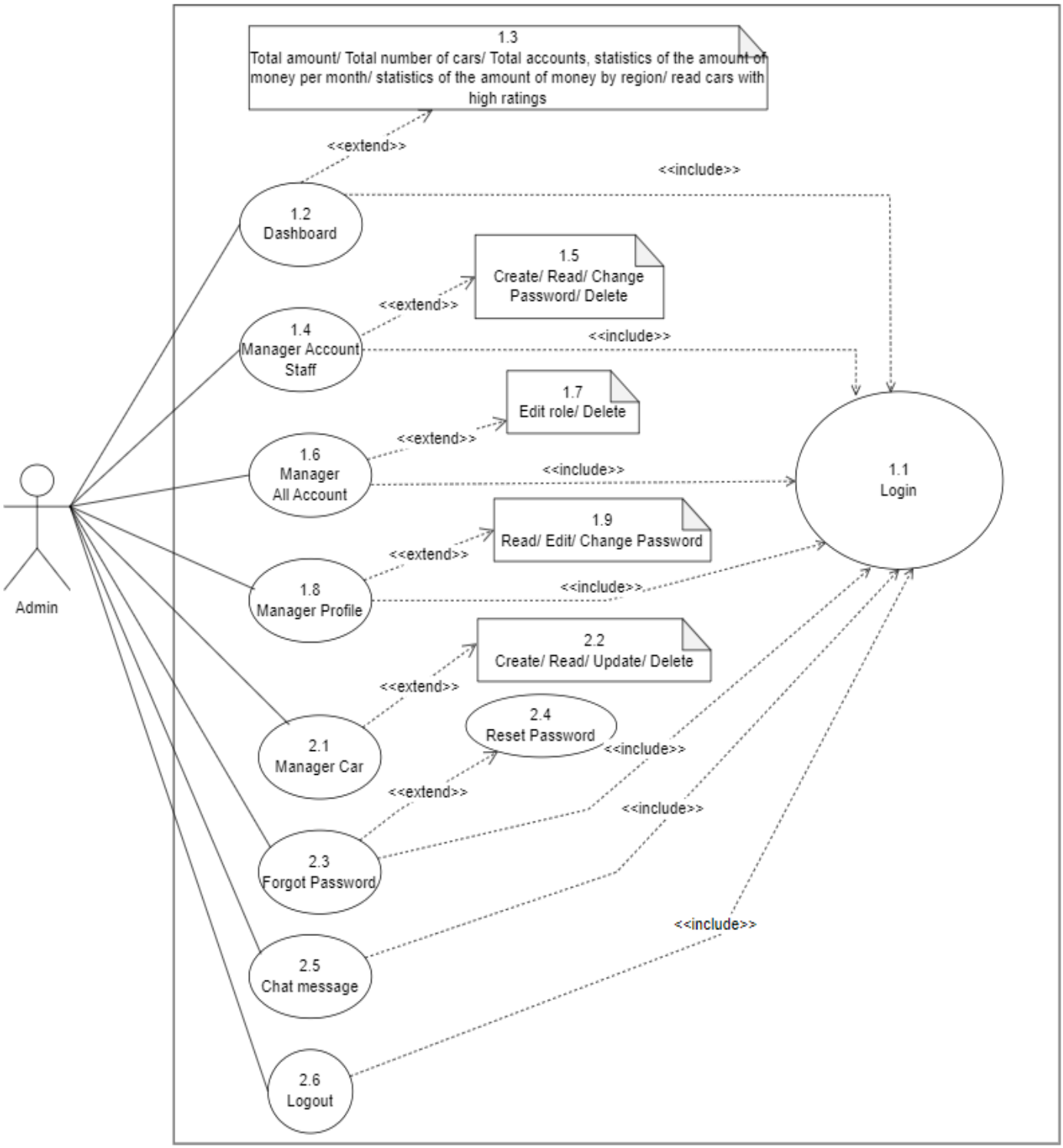



Figure 8: Use case diagram for admin.

**Primary Use-case scenario 1:**

| | |
|---|---|
| Use case ID | UC-1.1 |
| Use case name | Login |
| Actors | Admin |
| Pre-Condition(s) | • User must be Admin.<br>• Separate account is preferred.<br>• The admin account has been authorized.<br>• The admin device has been connected to the internet when performing login. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin enters password account or login with Google or login with Facebook with an existing admin account.<br>2. The system verifies the successful login and allows the administrator to access the application.<br>3. Admin goes to the dashboard page. | |
| Post-Condition(s) | • Admin successfully logged into the system. |

**Possible secondary scenario which could occur:**

- User entered incorrect login information.
- Google and Facebook of 3rd party software do not provide login permission for TCAR app anymore.
- The account role is not admin.

**Primary Use-case scenario 2:**

| | |
|---|---|
| Use case ID | UC-1.2 |
| Use case name | Dashboard |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want see dashboard page.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin log in to the system.<br>2. The system automatically redirects to the dashboard. | |
| Post-Condition(s) | • Admin see information of dashboard page. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 3:**

| Use case ID | UC-1.3 |
|---|---|
| Use case name | Total amount/ Total number of cars/ Total accounts, statistics of the amount of money per month/ statistics of the amount of money by region/ read cars with high ratings |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to see total amount, Total number of cars, Total accounts, statistics of the amount of money per month, statistics of the amount of money by region, read cars with high ratings<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin log in to the system.<br>2. The system automatically redirects to the dashboard.<br>3. Admin see total amount, total number of cars, total accounts, statistics of the amount of money per month, statistics of the amount of money by region, read cars with high ratings. | |
| Post-Condition(s) | • Admin see total amount, total number of cars, total accounts, statistics of the amount of money per month, statistics of the amount of money by region, read cars with high ratings. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 4:**

| Use case ID | UC-1.4 |
|---|---|
| Use case name | Manager Account Staff |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to manage the Staff account.<br>• User must be Admin. |

| | |
|---|---|
| **Flow of events:**<br>**Primary scenario**<br>1. Admin login to the system.<br>2. The system validates login information successfully and allows the admin to access the application.<br>3. Admin click on the button “Manager account Staff” to see all staff account. | |
| Post-Condition(s) | • Admin manages the account of Staff. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 5:**

| | |
|---|---|
| Use case ID | UC-1.5 |
| Use case name | Create (Create Account Staff) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to create the Staff account.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage account staff" button, the system will go to view all account staff page.<br>3. Admin clicks "Create a new account" button to create an account.<br>4. Admin enters all the information in the data form to create an account.<br>5. Create an account successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully created a Staff account.<br>• The data is saved to the database. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Email information authentication system already exists and requires creating a new email that does not match an existing email.
- Wrong data entered in the form.

**Primary Use-case scenario 6:**

| | |
|---|---|
| Use case ID | UC-1.5 |
| Use case name | Read (Read Account Staff) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to view all the Staff account.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage account staff" button, the system will go to view all account staff page.<br>3. Admin view all account staff. | |
| Post-Condition(s) | • Admin successfully viewed all staff account. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 7:**

| | |
|---|---|
| Use case ID | UC-1.5 |
| Use case name | Change password (Change password account staff) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to change password the Staff account.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage account staff" button, the system will go to view all account staff page.<br>3. Admin clicks "Change password" button (if existing user) to change password account.<br>4. Admin enters all information of new password in the data form to change password account.<br>5. Change password account successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully changed Staff password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.

- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Admin who enters the confirm password that do not match the new password will not be able to change the password.
- The Staff account that does not exist.

**Primary Use-case scenario 8:**

| Use case ID | UC-1.5 |
|---|---|
| Use case name | Delete (Delete account staff) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to delete the Staff account.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage account staff" button, the system will go to view all account staff page.<br>3. Admin clicks "Delete" button (if existing user) to delete account. | |
| Post-Condition(s) | • Admin successfully deleted Staff account (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- The Staff account that does not exist.

**Primary Use-case scenario 9:**

| Use case ID | UC-1.6 |
|---|---|
| Use case name | Manage All Account |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to manage all account.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin login to the system.<br>2. The system validates login information successfully and allows the admin to access the application. | |

| | |
|---|---|
| 3. Admin click on the button “Manager all account” to see all account. | |
| Post-Condition(s) | • Admin successfully deleted Staff password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 10:**

| | |
|---|---|
| Use case ID | UC-1.7 |
| Use case name | Edit role (Edit role all account) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to edit role all account.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage all account" button, the system will go to view all account page.<br>3. Admin clicks "Edit role" button (if existing user) to edit role for user.<br>4. Admin selects new role in the dropdown form to edit role for user.<br>5. Edit role for user successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully edited role for user (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- The account that does not exist.
- Admin did not select the role change value for the user.

**Primary Use-case scenario 11:**

| | |
|---|---|
| Use case ID | UC-1.7 |
| Use case name | Delete (Delete all account) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to delete any account. |

| | |
|---|---|
| | • User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage all account" button, the system will go to view all account page.<br>3. Admin clicks "Delete" button (if existing user) to delete account. | |
| Post-Condition(s) | • Admin successfully deleted account (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- The account that does not exist.

**Primary Use-case scenario 12:**

| | |
|---|---|
| Use case ID | UC-1.8 |
| Use case name | Manage Profile |
| Actors | Admin |
| Pre-Condition(s) | • Admin must login to the system if admin manages their profile.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin login to the system.<br>2. The system validates login information successfully and allows the admin to access the application.<br>3. Admin click on the button “Manager profile” to manage their profile. | |
| Post-Condition(s) | • Admin see their profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 13:**

| | |
|---|---|
| Use case ID | UC-1.9 |
| Use case name | Read (Read my profile) |
| Actors | Admin |

| | |
|---|---|
| Pre-Condition(s) | • Admin must be logged into the system if admin want to see their profile.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "My profile" button, the system will go to view their profile.<br>3. Admin view their profile. | |
| Post-Condition(s) | • Admin successfully viewed their profile. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 14:**

| | |
|---|---|
| Use case ID | UC-1.9 |
| Use case name | Edit (Update my profile) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to update their profile.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Edit profile" button, the system will go to update profile page.<br>4. Admin enters all information in the data form to update their profile.<br>5. Update profile successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully updated profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Admin does not enter data in the form.

**Primary Use-case scenario 15:**

| | |
|---|---|
| Use case ID | UC-1.9 |

| Use case name | Change password |
|---|---|
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to change their password.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Change password" button, the system will go to change password page.<br>3. Admin enters all information of new password in the data form to change password.<br>5. Change password successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully changed password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Admin does not enter data in the form.
- Admin who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 16:**

| Use case ID | UC-2.1 |
|---|---|
| Use case name | Manager Car |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to manage car.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin login to the system.<br>2. The system validates login information successfully and allows the admin to access the application.<br>3. Admin click on the button “Manager car” to see all car (is existing). | |
| Post-Condition(s) | • Admin manages all car. |

**Possible secondary scenario which could occur:**

- Login session expired.

- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 17:**

| | |
|---|---|
| Use case ID | UC-2.2 |
| Use case name | Create (Create car) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to create a new car.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage car" button, the system will go to view all car page.<br>3. Admin clicks "Create a new car" button to create a car.<br>4. Admin enters all the information in the data form to create a new car.<br>5. Create a new car successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully created a new car.<br>• The data is saved to the database. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Car information authentication system already exists and requires creating a new car that does not match an existing car.
- Wrong data entered in the form.

**Primary Use-case scenario 18:**

| | |
|---|---|
| Use case ID | UC-2.2 |
| Use case name | Read (Read all car) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to view all car).<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario** | |

| | |
|---|---|
| 1. Admin must log in to the system.<br>2. Admin clicks "Manage car" button, the system will go to view all car page.<br>3. Admin view all car. | |
| Post-Condition(s) | • Admin successfully viewed all car. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 19:**

| | |
|---|---|
| Use case ID | UC-2.2 |
| Use case name | Update (Update car) |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to update car.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage car" button, the system will go to view all car page.<br>3. Admin clicks "Update" button to create a car.<br>4. Admin enters all the information in the data form to update new car.<br>5. Update car successfully and the system saves data to the database. | |
| Post-Condition(s) | • Admin successfully updated car (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Car that does not exist.
- Wrong data entered in the form.

**Primary Use-case scenario 20:**

| | |
|---|---|
| Use case ID | UC-2.2 |
| Use case name | Delete (Delete car) |
| Actors | Admin |

| | |
|---|---|
| Pre-Condition(s) | • Admin must be logged into the system if admin want to delete car.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Manage car" button, the system will go to view all car.<br>3. Admin clicks "Delete" button (if existing car) to delete car. | |
| Post-Condition(s) | • Admin successfully deleted car (if existing car) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.
- Car that does not exist.

**Primary Use-case scenario 21:**

| | |
|---|---|
| Use case ID | UC-2.3 |
| Use case name | Forgot Password |
| Actors | Admin |
| Pre-Condition(s) | • User must access the application. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin go to login page.<br>2. Admin clicks "Forgot password" button, the system will go to forgot password page.<br>3. Admin enters all the information in the data form of forgot password page. | |
| Post-Condition(s) | • Admin go to password reset page. |

**Possible secondary scenario which could occur:**

- Account that does not exist.
- Wrong data entered in the form.
- User does not receive email or SMS.

**Primary Use-case scenario 22:**

| | |
|---|---|
| Use case ID | UC-2.4 |
| Use case name | Reset Password |
| Actors | Admin |
| Pre-Condition(s) | • User must access the application. |

| | |
|---|---|
| | • Admin is received link reset password form email or OTP from SMS. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin receive link reset password form email or OTP from SMS.<br>Method 1: Link reset password form email.<br>2. User go to email and click link in email.<br>3. User enters all the information in the data form to reset password.<br>Method 2: OTP from SMS.<br>2. User get OTP code from phone and enter OTP to app.<br>3. User enters all the information in the data form to reset password. | |
| Post-Condition(s) | • Admin successfully reset password. |

**Possible secondary scenario which could occur:**

- User does not enter data in the form.
- User who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 23:**

| | |
|---|---|
| Use case ID | UC-2.5 |
| Use case name | Chat message |
| Actors | Admin |
| Pre-Condition(s) | • Admin must be logged into the system if admin want to chat message.<br>• User must be Admin. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Message" button, the system will go to chat message page.<br>3. Admin can find people, create groups, rename groups, add people, delete people in the group or leave the group.<br>4. Admin message with people who have permission in the system. | |
| Post-Condition(s) | • Admin chat message. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.

- The system cannot find other users.

**Primary Use-case scenario 24:**

| | |
|---|---|
| Use case ID | UC-2.6 |
| Use case name | Logout |
| Actors | Admin |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. Admin must log in to the system.<br>2. Admin clicks "Logout" button, the system will logout and go to home page. | |
| Post-Condition(s) | • Admin successfully logout. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the admin account is not logged in correctly and is unable to access the application.

### 8.4.2. Use case diagram, primary and secondary use-case scenario for staff:

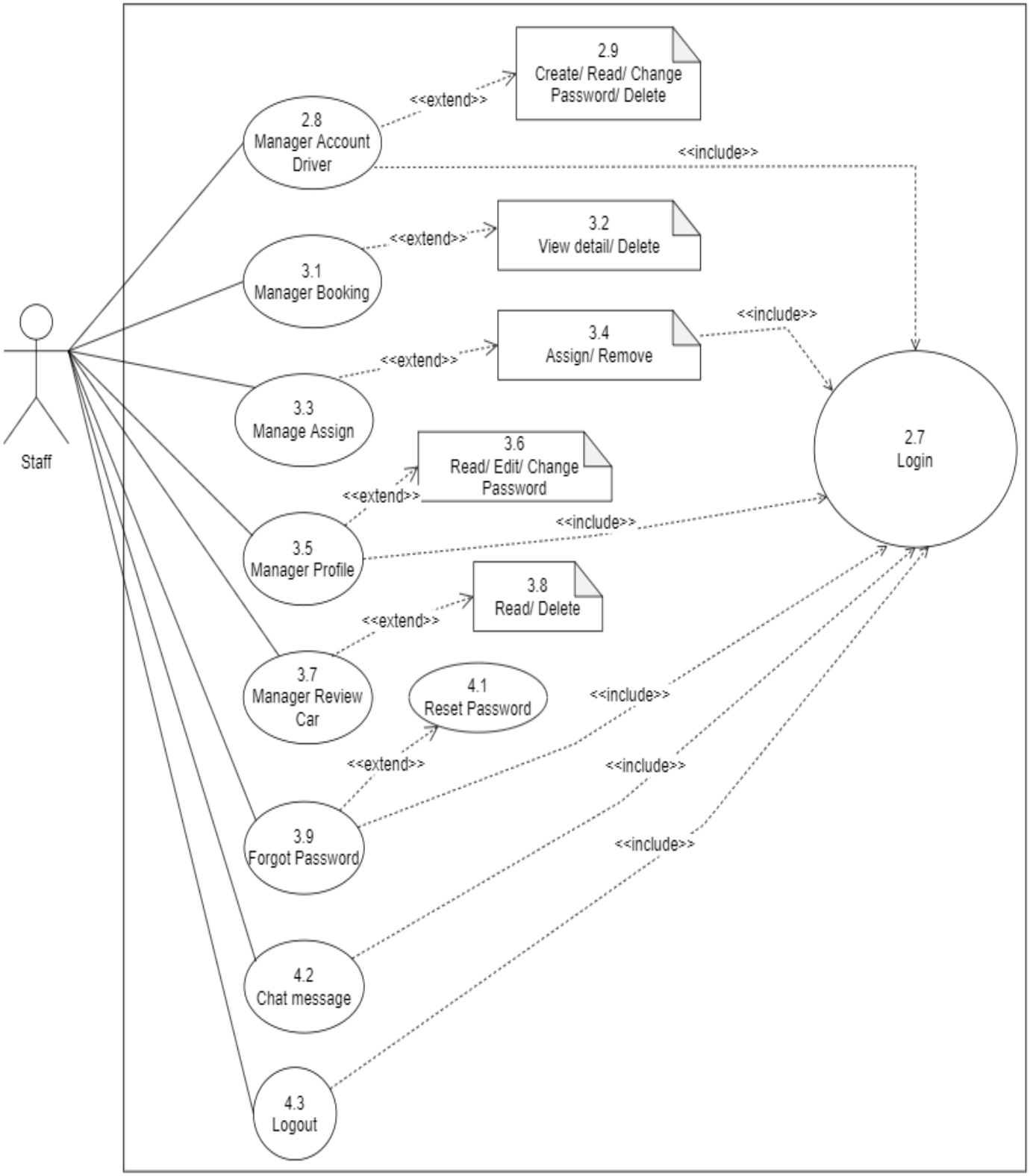



Figure 9: Use case diagram for staff.

**Primary Use-case scenario 1:**

| Use case ID | UC-2.7 |
|---|---|
| Use case name | Login |
| Actors | Staff |
| Pre-Condition(s) | • User must be staff.<br>• Separate account is preferred.<br>• The staff account has been authorized.<br>• The staff device has been connected to the internet when performing login. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff enters password account or login with Google or login with Facebook with an existing driver account.<br>2. The system verifies the successful login and allows the staff to access the application.<br>3. Staff goes to the dashboard page. | |
| Post-Condition(s) | • Staff successfully logged into the system. |

**Possible secondary scenario which could occur:**

- User entered incorrect login information.
- Google and Facebook of 3rd party software do not provide login permission for TCAR app anymore.
- The account role is not staff.

**Primary Use-case scenario 2:**

| Use case ID | UC-2.8 |
|---|---|
| Use case name | Manager Account Driver |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to manage the driver account.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. Staff click on the button “Manager account Driver” to see all staff account. | |
| Post-Condition(s) | • Staff manages the account of Driver. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 3:**

| | |
|---|---|
| Use case ID | UC-2.9 |
| Use case name | Create (Create Account Driver) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to create the driver account.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account driver" button, system will go to view all account driver page.<br>3. Staff clicks "Create a new account" button to create an account.<br>4. Staff enters all the information in the data form to create an account.<br>5. Create an account successfully and the system saves data to the database. | |
| Post-Condition(s) | • Staff successfully created a driver account.<br>• The data is saved to the database. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Email information authentication system already exists and requires creating a new email that does not match an existing email.
- Wrong data entered in the form.

**Primary Use-case scenario 4:**

| | |
|---|---|
| Use case ID | UC-2.9 |
| Use case name | Read (Read Account Driver) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to view all the driver account. |

| | |
|---|---|
| | • User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account driver" button, system will go to view all account driver page.<br>3. Staff view all account driver. | |
| Post-Condition(s) | • Staff successfully viewed all driver account. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 5:**

| | |
|---|---|
| Use case ID | UC-2.9 |
| Use case name | Change password (Change password account driver) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to change password the driver account.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account driver" button, system will go to view all account driver page.<br>3. Staff clicks "Change password" button (if existing user) to change password account.<br>4. Staff enters all information of new password in the data form to change password account.<br>5. Change password account successfully and the system saves data to the database. | |
| Post-Condition(s) | • Staff successfully changed driver password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Staff who enters the confirm password that do not match the new password will not be able to change the password.
- The driver account that does not exist.

**Primary Use-case scenario 6:**

| Use case ID | UC-2.9 |
|---|---|
| Use case name | Delete (Delete account driver) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to delete the driver account.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account driver" button, the system will go to view all account driver page.<br>3. Staff clicks "Delete" button (if existing user) to delete account. | |
| Post-Condition(s) | • Staff successfully deleted driver account (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- The driver account that does not exist.

**Primary Use-case scenario 7:**

| Use case ID | UC-3.1 |
|---|---|
| Use case name | Manager Booking |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to manage the booking.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. Staff click on the button “Manager all booking” to see all booking. | |
| Post-Condition(s) | • Staff manages all booking. |

**Possible secondary scenario which could occur:**

- Login session expired.

- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 8:**

| Use case ID | UC-3.2 |
|---|---|
| Use case name | View detail (View detail booking) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to view detail booking.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. Staff click on the button “Manager all booking” to see all booking.<br>3. Staff click on the button “View” to view detail booking. | |
| Post-Condition(s) | • Staff view detail booking. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 9:**

| Use case ID | UC-3.2 |
|---|---|
| Use case name | Delete (Delete booking) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to delete booking.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account booking" button, the system will go to view all booking page.<br>3. Staff clicks "Delete" button (if existing booking) to delete booking. | |

| | |
|---|---|
| Post-Condition(s) | • Staff successfully deleted booking (if existing booking). |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Booking that does not exist.

**Primary Use-case scenario 10:**

| | |
|---|---|
| Use case ID | UC-3.1 |
| Use case name | Manager Assign |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to manage assign.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. Staff click on the button “Assign car” to see all car assign or not assign. | |
| Post-Condition(s) | • Staff manages all car assign or not assign. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 11:**

| | |
|---|---|
| Use case ID | UC-3.4 |
| Use case name | Assign (Assign driver to car) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to assign driver to car.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario** | |

| | |
|---|---|
| 1. Staff must log in to the system.<br>2. Staff clicks "Assign car" button, the system will go to view all car assign and not assign.<br>3. Staff clicks "Assign" button (if existing car) to go to view all user does not assign car.<br>4. Staff clicks "Assign this driver to car" button (if existing driver) to go to assign car to driver. | |
| Post-Condition(s) | • Staff successfully assigned car. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Car or driver that does not exist.
- Driver has already been assigned to another car, or the assigned car and driver are not in the same location.

**Primary Use-case scenario 12:**

| | |
|---|---|
| Use case ID | UC-3.4 |
| Use case name | Remove (Remove assign driver to car) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to remove assign driver to car.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Assign car" button, the system will go to view all car assign and not assign.<br>3. Staff clicks "Remove" button (if existing assigned driver) to remove assign driver to car. | |
| Post-Condition(s) | • Staff successfully removed assign car. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Car or driver that does not exist.
- Car is not assigned to driver.

**Primary Use-case scenario 13:**

| | |
|---|---|
| Use case ID | UC-3.5 |
| Use case name | Manage Profile |

| Actors | Staff |
|---|---|
| Pre-Condition(s) | • Staff must login to the system if staff manages their profile.<br>• User must be staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. Staff click on the button “Manager profile” to manage their profile. | |
| Post-Condition(s) | • Staff see their profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 14:**

| Use case ID | UC-3.6 |
|---|---|
| Use case name | Read (Read my profile) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to see their profile.<br>• User must be staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "My profile" button, the system will go to view their profile.<br>3. Staff view their profile. | |
| Post-Condition(s) | • Staff successfully viewed their profile. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 15:**

| Use case ID | UC-3.6 |
|---|---|
| Use case name | Edit (Update my profile) |

| Actors | Staff |
|---|---|
| Pre-Condition(s) | • Staff must be logged into the system if staff want to update their profile.<br>• User must be staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Edit profile" button, the system will go to update profile page.<br>4. Staff enters all information in the data form to update their profile.<br>5. Update profile successfully and the system saves data to the database. | |
| Post-Condition(s) | • Staff successfully updated profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Staff does not enter data in the form.

**Primary Use-case scenario 16:**

| Use case ID | UC-3.6 |
|---|---|
| Use case name | Change password |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to change their password.<br>• User must be staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Change password" button, the system will go to change password page.<br>3. Staff enters all information of new password in the data form to change password.<br>5. Change password successfully and the system saves data to the database. | |
| Post-Condition(s) | • Staff successfully changed password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Staff does not enter data in the form.

- Staff who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 17:**

| | |
|---|---|
| Use case ID | UC-3.7 |
| Use case name | Manager Review Car |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to review car.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. Staff click on the button “Manager review” to see all review car. | |
| Post-Condition(s) | • Staff manages all review car. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 18:**

| | |
|---|---|
| Use case ID | UC-3.8 |
| Use case name | Read (Read review detail) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to read review detail.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account driver" button, system will go to view all car have review.<br>3. Staff clicks "View detail" button, system will go to view detail of review for that car.<br>3. Staff view review detail. | |
| Post-Condition(s) | • Staff view review detail. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- No data in database.

**Primary Use-case scenario 19:**

| | |
|---|---|
| Use case ID | UC-3.8 |
| Use case name | Delete (Delete review) |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to delete review.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Manage account driver" button, system will go to view all car have review.<br>3. Staff clicks "View detail" button, system will go to view detail of review for that car.<br>3. Staff clicks "Delete" button (if existing review) to delete review. | |
| Post-Condition(s) | • Staff successfully deleted review (if existing review) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Review that does not exist.

**Primary Use-case scenario 20:**

| | |
|---|---|
| Use case ID | UC-3.9 |
| Use case name | Forgot Password |
| Actors | Staff |
| Pre-Condition(s) | • User must access the application. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff go to login page.<br>2. Staff clicks "Forgot password" button, the system will go to forgot password page.<br>3. Staff enters all the information in the data form of forgot password page. | |
| Post-Condition(s) | • Staff go to password reset page. |

**Possible secondary scenario which could occur:**

- Account that does not exist.
- Wrong data entered in the form.
- User does not receive email or SMS.

**Primary Use-case scenario 21:**

| | |
|---|---|
| Use case ID | UC-4.1 |
| Use case name | Reset Password |
| Actors | Staff |
| Pre-Condition(s) | • User must access the application.<br>• Staff is received link reset password form email or OTP from SMS. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff receive link reset password form email or OTP from SMS.<br>Method 1: Link reset password form email.<br>2. User go to email and click link in email.<br>3. User enters all the information in the data form to reset password.<br>Method 2: OTP from SMS.<br>2. User get OTP code from phone and enter OTP to app.<br>3. User enters all the information in the data form to reset password. | |
| Post-Condition(s) | • Staff successfully reset password |

**Possible secondary scenario which could occur:**

- User does not enter data in the form.
- User who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 22:**

| | |
|---|---|
| Use case ID | UC-4.2 |
| Use case name | Chat message |
| Actors | Staff |
| Pre-Condition(s) | • Staff must be logged into the system if staff want to chat message.<br>• User must be staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Message" button, the system will go to chat message page. | |

| | |
|---|---|
| 3. Staff can find people, create groups, rename groups, add people, delete people in the group or leave the group.<br>4. Staff message with people who have permission in the system. | |
| Post-Condition(s) | • Staff chat message. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- The system cannot find other users.

**Primary Use-case scenario 24:**

| | |
|---|---|
| Use case ID | UC-4.3 |
| Use case name | Logout |
| Actors | Staff |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. Staff must log in to the system.<br>2. Staff clicks "Logout" button, the system will logout and go to home page. | |
| Post-Condition(s) | • Staff successfully logout. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.

### 8.4.3. Use case diagram, primary and secondary use-case scenario for driver:

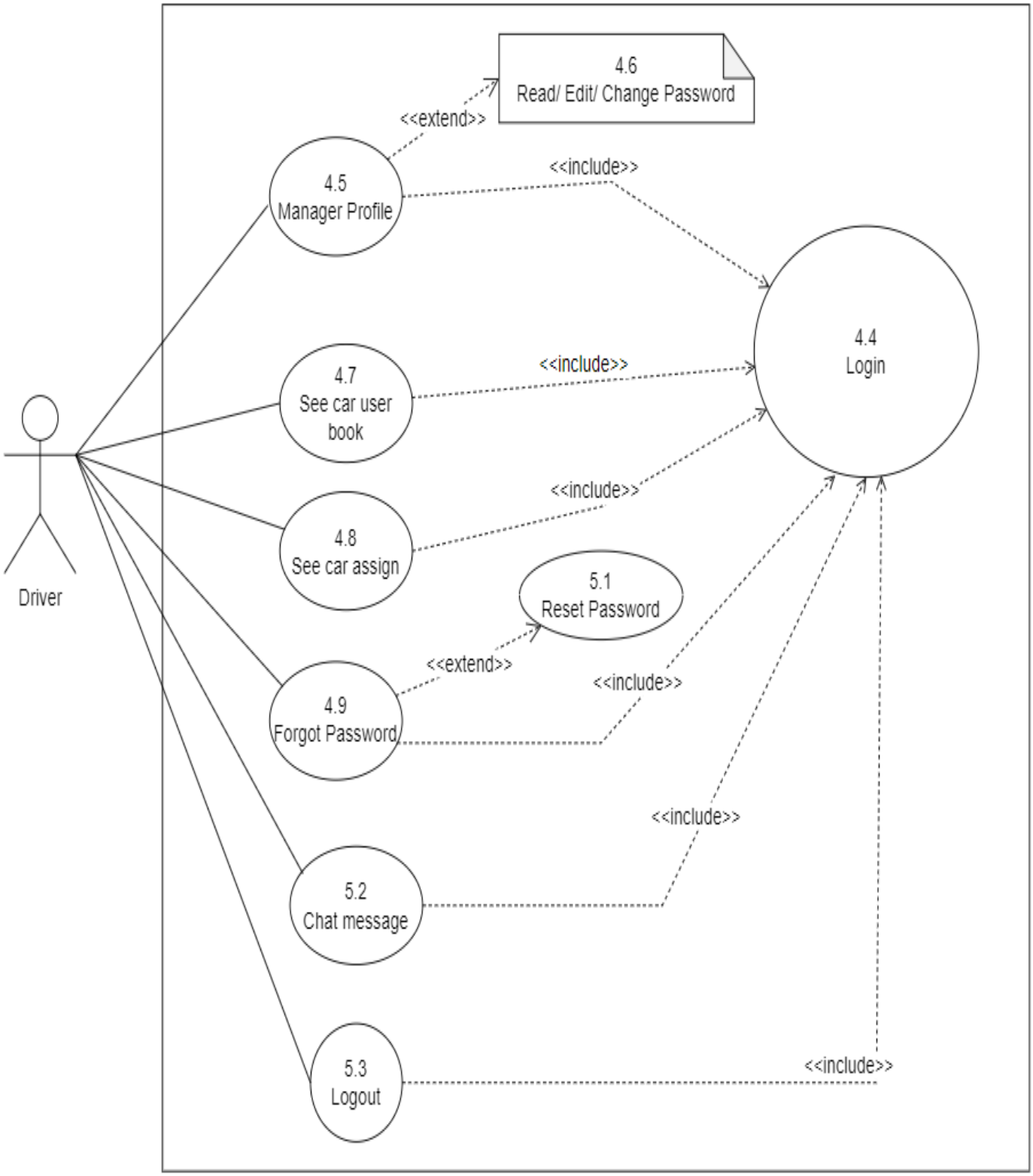



Figure 10: Use case diagram for driver.

**Primary Use-case scenario 1:**

| | |
|---|---|
| Use case ID | UC-4.4 |
| Use case name | Login |
| Actors | Driver |
| Pre-Condition(s) | • User must be driver.<br>• Separate account is preferred.<br>• The driver account has been authorized.<br>• The driver device has been connected to the internet when performing login. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver enters password account or login with Google or login with Facebook with an existing driver account.<br>2. The system verifies the successful login and allows the driver to access the application.<br>3. Driver goes to the dashboard page. | |
| Post-Condition(s) | • Driver successfully logged into the system. |

**Possible secondary scenario which could occur:**

- User entered incorrect login information.
- Google and Facebook of 3rd party software do not provide login permission for TCAR app anymore.
- The account role is not driver.

**Primary Use-case scenario 2:**

| | |
|---|---|
| Use case ID | UC-4.5 |
| Use case name | Manage Profile |
| Actors | Driver |
| Pre-Condition(s) | • Driver must login to the system if staff manages their profile.<br>• User must be Driver. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver login to the system.<br>2. The system validates login information successfully and allows the driver to access the application.<br>3. Driver click on the button “Manager profile” to manage their profile. | |
| Post-Condition(s) | • Driver sees their profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 3:**

| | |
|---|---|
| Use case ID | UC-4.6 |
| Use case name | Read (Read my profile) |
| Actors | Driver |
| Pre-Condition(s) | • Driver must be logged into the system if driver want to see their profile.<br>• User must be driver. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver must log in to the system.<br>2. Driver clicks "My profile" button, the system will go to view their profile.<br>3. Driver f view their profile. | |
| Post-Condition(s) | • Driver successfully viewed their profile. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 4:**

| | |
|---|---|
| Use case ID | UC-4.6 |
| Use case name | Edit (Update my profile) |
| Actors | Driver |
| Pre-Condition(s) | • Driver must be logged into the system if driver want to update their profile.<br>• User must be staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver must log in to the system.<br>2. Driver clicks "Edit profile" button, the system will go to update profile page.<br>4. Driver enters all information in the data form to update their profile.<br>5. Update profile successfully and the system saves data to the database. | |
| Post-Condition(s) | • Driver successfully updated profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- Driver does not enter data in the form.

**Primary Use-case scenario 5:**

| Use case ID | UC-4.6 |
|---|---|
| Use case name | Change password |
| Actors | Driver |
| Pre-Condition(s) | • Driver must be logged into the system if driver want to change their password.<br>• User must be driver. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver must log in to the system.<br>2. Driver clicks "Change password" button, the system will go to change password page.<br>3. Driver enters all information of new password in the data form to change password.<br>5. Change password successfully and the system saves data to the database. | |
| Post-Condition(s) | • Driver successfully changed password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- Driver does not enter data in the form.
- Driver who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 6:**

| Use case ID | UC-4.7 |
|---|---|
| Use case name | See car user book |
| Actors | Driver |
| Pre-Condition(s) | • Driver must be logged into the system if driver want to see car user book.<br>• User must be Staff. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver login to the system. | |

| | |
|---|---|
| 2. The system validates login information successfully and allows the driver to access the application.<br>3. Driver click on the button “My car user book” to see car user book.<br>4. Driver see car user book. | |
| Post-Condition(s) | • Driver sees car user book. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- No data in database.
- This driver has not been booked by any user yet.

**Primary Use-case scenario 7:**

| | |
|---|---|
| Use case ID | UC-4.8 |
| Use case name | See car assign |
| Actors | Driver |
| Pre-Condition(s) | • Driver must be logged into the system if driver want to see car assign.<br>• User must be driver. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver login to the system.<br>2. The system validates login information successfully and allows the driver to access the application.<br>3. Driver click on the button “My car assign” to see car assign.<br>4. Driver see car assign. | |
| Post-Condition(s) | • Driver sees car user book.<br>• Staff assign car to driver. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- No data in database.
- This driver has not been assigned by any car yet.

**Primary Use-case scenario 8:**

| Use case ID | UC-3.9 |
|---|---|
| Use case name | Forgot Password |
| Actors | Driver |
| Pre-Condition(s) | • User must access the application. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver go to login page.<br>2. Driver clicks "Forgot password" button, the system will go to forgot password page.<br>3. Driver enters all the information in the data form of forgot password page. | |
| Post-Condition(s) | • Driver go to password reset page. |

**Possible secondary scenario which could occur:**

- Account that does not exist.
- Wrong data entered in the form.
- User does not receive email or SMS.

**Primary Use-case scenario 9:**

| Use case ID | UC-4.1 |
|---|---|
| Use case name | Reset Password |
| Actors | Driver |
| Pre-Condition(s) | • User must access the application.<br>• Driver is received link reset password form email or OTP from SMS. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver receive link reset password form email or OTP from SMS.<br>Method 1: Link reset password form email.<br>2. User go to email and click link in email.<br>3. User enters all the information in the data form to reset password.<br>Method 2: OTP from SMS.<br>2. User get OTP code from phone and enter OTP to app.<br>3. User enters all the information in the data form to reset password. | |
| Post-Condition(s) | • Driver successfully reset password. |

**Possible secondary scenario which could occur:**

- User does not enter data in the form.
- User who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 10:**

| | |
|---|---|
| Use case ID | UC-4.2 |
| Use case name | Chat message |
| Actors | Driver |
| Pre-Condition(s) | • Driver must be logged into the system if driver want to chat message.<br>• User must be driver. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver must log in to the system.<br>2. Driver clicks "Message" button, the system will go to chat message page.<br>3. Driver can find people, create groups, rename groups, add people, delete people in the group or leave the group.<br>4. Driver message with people who have permission in the system. | |
| Post-Condition(s) | • Driver chat message. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- The system cannot find other users.

**Primary Use-case scenario 11:**

| | |
|---|---|
| Use case ID | UC-4.3 |
| Use case name | Logout |
| Actors | Driver |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver must log in to the system.<br>2. Driver clicks "Logout" button, the system will logout and go to home page. | |
| Post-Condition(s) | • Driver successfully logout. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.

### 8.4.4. Use case diagram, primary and secondary use-case scenario for user:

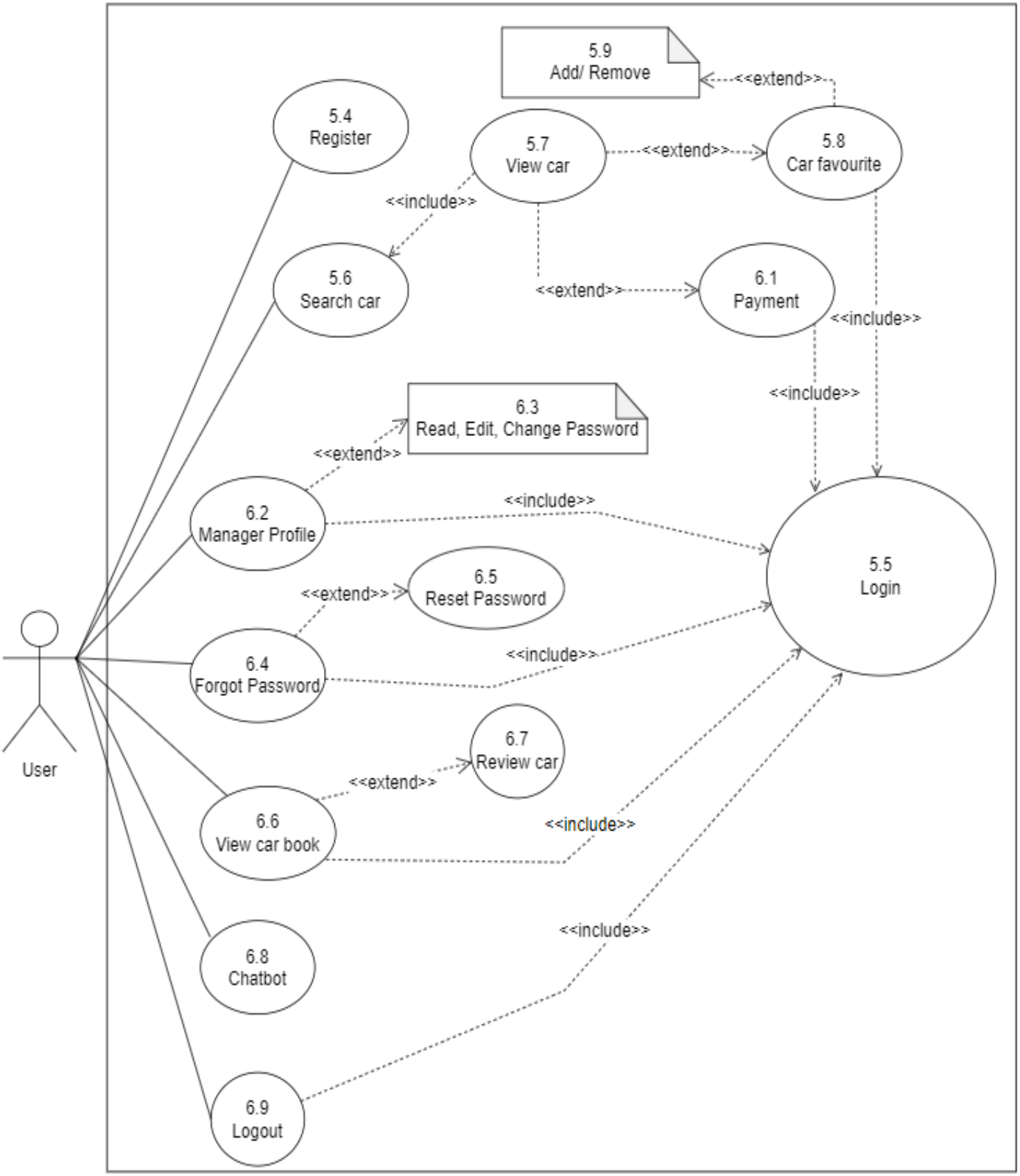



Figure 11: Use case diagram for user.

**Primary Use-case scenario 1:**

| | |
|---|---|
| Use case ID | UC-5.4 |
| Use case name | Register |
| Actors | User |
| Pre-Condition(s) | • User must use real email.<br>• The admin device has been connected to the internet when performing register. |
| **Flow of events:**<br>**Primary scenario**<br>1. User clicks "Register" button, system will go to register page.<br>2. Driver enters all information in the data form to register account.<br>3. User go to email to get link active account.<br>4. User go to home page. | |
| Post-Condition(s) | • User successfully registers. |

**Possible secondary scenario which could occur:**

- User entered incorrect information form register.
- User cannot receive email.

**Primary Use-case scenario 2:**

| | |
|---|---|
| Use case ID | UC-5.5 |
| Use case name | Login |
| Actors | User |
| Pre-Condition(s) | • Separate account is preferred.<br>• The admin account has been authorized.<br>• The admin device has been connected to the internet when performing login. |
| **Flow of events:**<br>**Primary scenario**<br>1. User enters password account or login with Google or login with Facebook with an existing account.<br>2. The system verifies the successful login and allows user to access the application.<br>3. User goes to the dashboard page. | |
| Post-Condition(s) | • User successfully logged into the system. |

**Possible secondary scenario which could occur:**

- User entered incorrect login information.

- Google and Facebook of 3rd party software do not provide login permission for TCAR app anymore.

**Primary Use-case scenario 3:**

| | |
|---|---|
| Use case ID | UC-5.6 |
| Use case name | Search car |
| Actors | User |
| Pre-Condition(s) | • User access application. |
| **Flow of events:**<br>**Primary scenario**<br>1. User go to home page.<br>2. The user search by start day, end day, seats category, location, and so on.<br>3. User go to car page. | |
| Post-Condition(s) | • User can login or not login into the system. |

**Possible secondary scenario which could occur:**

- User entered incorrect information form search.

**Primary Use-case scenario 4:**

| | |
|---|---|
| Use case ID | UC-5.7 |
| Use case name | View car |
| Actors | User |
| Pre-Condition(s) | • User access application. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver can search car or click “Car” button or chatbot support find car.<br>2. Driver see all car or filter car. | |
| Post-Condition(s) | • Driver see car. |

**Possible secondary scenario which could occur:**

- User entered incorrect information form search.
- Chatbot not support.

**Primary Use-case scenario 5:**

| | |
|---|---|
| Use case ID | UC-5.8 |
| Use case name | View car |
| Actors | User |
| Pre-Condition(s) | • User access application. |

| | |
|---|---|
| **Flow of events:**<br>**Primary scenario**<br>1. Driver can search car or click “Car” button or chatbot support find car.<br>2. Driver see all car or filter car. | |
| Post-Condition(s) | • Driver see car. |

**Possible secondary scenario which could occur:**

- User entered incorrect information form search.
- Chatbot not support.

**Primary Use-case scenario 6:**

| | |
|---|---|
| Use case ID | UC-5.8 |
| Use case name | Car favorite |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User login to the system.<br>2. User click on the button “My favorite car” to see the cars that user love. | |
| Post-Condition(s) | • User see favorite cars. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 8:**

| | |
|---|---|
| Use case ID | UC-5.9 |
| Use case name | Add (Add car to favorite car) |
| Actors | Staff |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User to car page.<br>3. Staff clicks "Add favorite car" button, car is added to favorite cart. | |
| Post-Condition(s) | • User successfully added car to favorite cart. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Car that does not exist.
- Driver has already been added to favorite cart.

**Primary Use-case scenario 9:**

| | |
|---|---|
| Use case ID | UC-5.9 |
| Use case name | Remove (Remove car to favorite car) |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User to car page.<br>3. Staff clicks "Remove" button (if existing added car) to remove favorite car. | |
| Post-Condition(s) | • User successfully removed favorite car. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 10:**

| | |
|---|---|
| Use case ID | UC-6.1 |
| Use case name | Payment |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system.<br>• Users must enter their full personal information.<br>• Users must use an official credit card. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User to car page.<br>3. User clicks "Book car" button and fill all information booking.<br>4. User payment. | |
| Post-Condition(s) | • User successfully payment. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.
- Users must use real credit card.

**Primary Use-case scenario 11:**

| | |
|---|---|
| Use case ID | UC-6.2 |
| Use case name | Manage Profile |
| Actors | User |
| Pre-Condition(s) | • User must login to the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User login to the system.<br>2. The system validates login information successfully and allows the staff to access the application.<br>3. User click on the button “Manager profile” to manage their profile. | |
| Post-Condition(s) | • User see their profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the user account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 12:**

| | |
|---|---|
| Use case ID | UC-6.3 |
| Use case name | Read (Read my profile) |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User clicks "My profile" button, the system will go to view their profile.<br>3. User view their profile. | |
| Post-Condition(s) | • User successfully viewed their profile. |

**Possible secondary scenario which could occur:**

- Login session expired.

- The system verifies that the user account is not logged in correctly and is unable to access the application.

**Primary Use-case scenario 13:**

| | |
|---|---|
| Use case ID | UC-6.3 |
| Use case name | Edit (Update my profile) |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User clicks "Edit profile" button, the system will go to update profile page.<br>4. User enters all information in the data form to update their profile.<br>5. Update profile successfully and the system saves data to the database. | |
| Post-Condition(s) | • User successfully updated profile (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that user account is not logged in correctly and is unable to access the application.
- User does not enter data in the form.

**Primary Use-case scenario 14:**

| | |
|---|---|
| Use case ID | UC-6.3 |
| Use case name | Change password |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User clicks "Change password" button, the system will go to change password page.<br>3. User enters all information of new password in the data form to change password.<br>5. Change password successfully and the system saves data to the database. | |
| Post-Condition(s) | • User successfully changed password (if existing user) |

**Possible secondary scenario which could occur:**

- Login session expired.

- The system verifies that user account is not logged in correctly and is unable to access the application.
- User does not enter data in the form.
- User who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 15:**

| | |
|---|---|
| Use case ID | UC-6.4 |
| Use case name | Forgot Password |
| Actors | User |
| Pre-Condition(s) | • User must access the application. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver go to login page.<br>2. Driver clicks "Forgot password" button, the system will go to forgot password page.<br>3. Driver enters all the information in the data form of forgot password page. | |
| Post-Condition(s) | • User go to password reset page. |

**Possible secondary scenario which could occur:**

- Account that does not exist.
- Wrong data entered in the form.
- User does not receive email or SMS.

**Primary Use-case scenario 16:**

| | |
|---|---|
| Use case ID | UC-6.5 |
| Use case name | Reset Password |
| Actors | User |
| Pre-Condition(s) | • User must access the application.<br>• User is received link reset password form email or OTP from SMS. |
| **Flow of events:**<br>**Primary scenario**<br>1. User receive link reset password form email or OTP from SMS.<br>Method 1: Link reset password form email.<br>2. User go to email and click link in email.<br>3. User enters all the information in the data form to reset password.<br>Method 2: OTP from SMS.<br>2. User get OTP code from phone and enter OTP to app. | |

| | |
|---|---|
| 3. User enters all the information in the data form to reset password. | |
| Post-Condition(s) | • User successfully reset password. |

**Possible secondary scenario which could occur:**

- User does not enter data in the form.
- User who enters the confirm password that do not match the new password will not be able to change the password.

**Primary Use-case scenario 17:**

| | |
|---|---|
| Use case ID | UC-6.6 |
| Use case name | View car book |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system.<br>• User must book car already. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver login to the system.<br>2. Driver click on the button “My car book” to see car book. | |
| Post-Condition(s) | • User see car book. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- No data in database.
- User must book car already.

**Primary Use-case scenario 18:**

| | |
|---|---|
| Use case ID | UC-6.7 |
| Use case name | Review car |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system.<br>• Car must be in the done status. |
| **Flow of events:**<br>**Primary scenario**<br>1. Driver login to the system.<br>2. Driver click on the button “My car book” to see car book.<br>3. Driver click on the button “Review car” to book car. | |

| | |
|---|---|
| Post-Condition(s) | • User review car. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- Car must be in the done status.
- User must book car already.

**Primary Use-case scenario 19:**

| | |
|---|---|
| Use case ID | UC-6.8 |
| Use case name | Chatbot |
| Actors | User |
| Pre-Condition(s) | • User access application. |
| **Flow of events:**<br>**Primary scenario**<br>1. User can login or need not login to the system.<br>2. User ask chatbot. | |
| Post-Condition(s) | • Chatbot active. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the driver account is not logged in correctly and is unable to access the application.
- Dialogflow stopped providing service.
- Chatbot must active.

**Primary Use-case scenario 20:**

| | |
|---|---|
| Use case ID | UC-6.9 |
| Use case name | Logout |
| Actors | User |
| Pre-Condition(s) | • User must be logged into the system. |
| **Flow of events:**<br>**Primary scenario**<br>1. User must log in to the system.<br>2. User clicks "Logout" button, the system will logout and go to home page. | |
| Post-Condition(s) | • User successfully logout. |

**Possible secondary scenario which could occur:**

- Login session expired.
- The system verifies that the staff account is not logged in correctly and is unable to access the application.

## 8.5. Entity relationship diagrams (ERD):

### 8.5.1. Physical design:

This is an ERD designed for the system to create, query, update, and delete data in the TCAR application. Collections will have blue frames and green frames will be documents related to ID from collections. This ERD have 5 collections is users, cars, bookings, chats, messages. Collections and documents both explicitly define the data type for each field. Collection users will be the focus for other collections to get ID for the purpose of performing necessary functions. Collection cars have 3 fields userCreateId, reviews and assign will have a one-to-many relationship with collection users. Collection bookings has a userBooking document with user fields that have a one-to-many relationship with collections users. In addition, the bookings collection also has a document bookCars with fields driverID that has a one-to-many relationship with colletion users, and the bookings collection also has a document bookCars with fields car that has a one-to-many relationship with colletion cars. Collection chats will have 2 fields users and groupCreatedBy has a one-to-many relationship with collections users. Collection chats will have id fields that have a one-to-many relationship with collection messages. Finally, the messages collection has an id field, and the sender sequence has a many-to-one relationship with collections chats and users respectively.

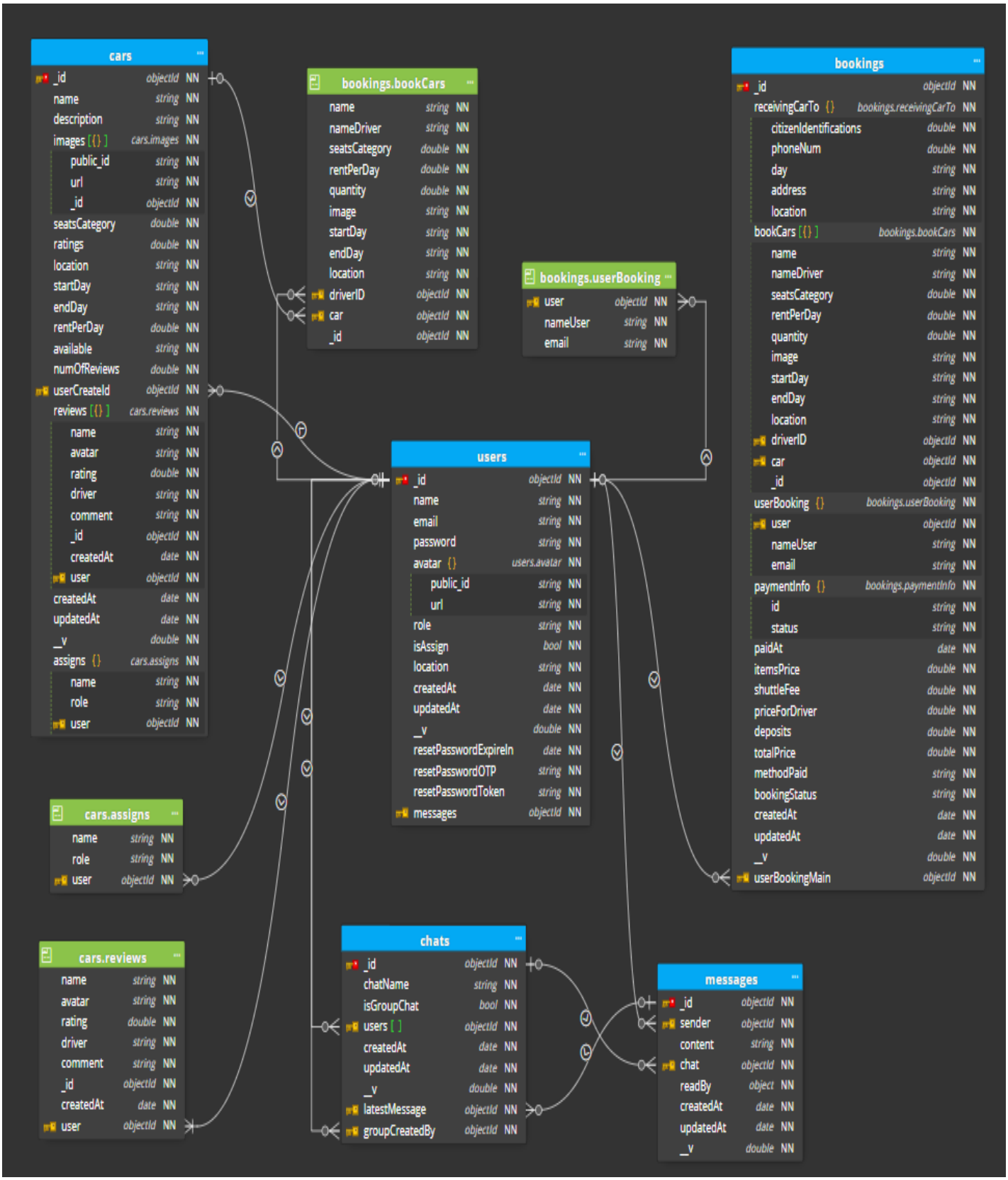



Table 5: ERD of TCAR app.

## 8.6. Wireframes for prototypes:

### 8.6.1. Navigation bar wireframes:

This is the navigation bar wireframe when the user is not logged in.

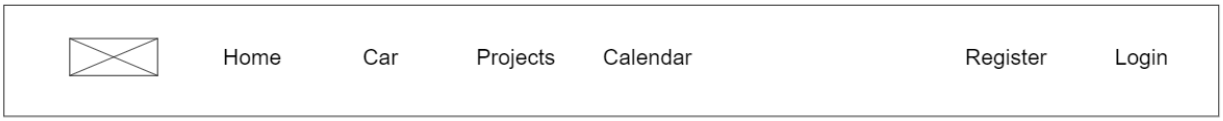



Figure 12: Navigation bar wireframe (not logged in).

This is the navigation bar wireframe when the user is login with admin role.

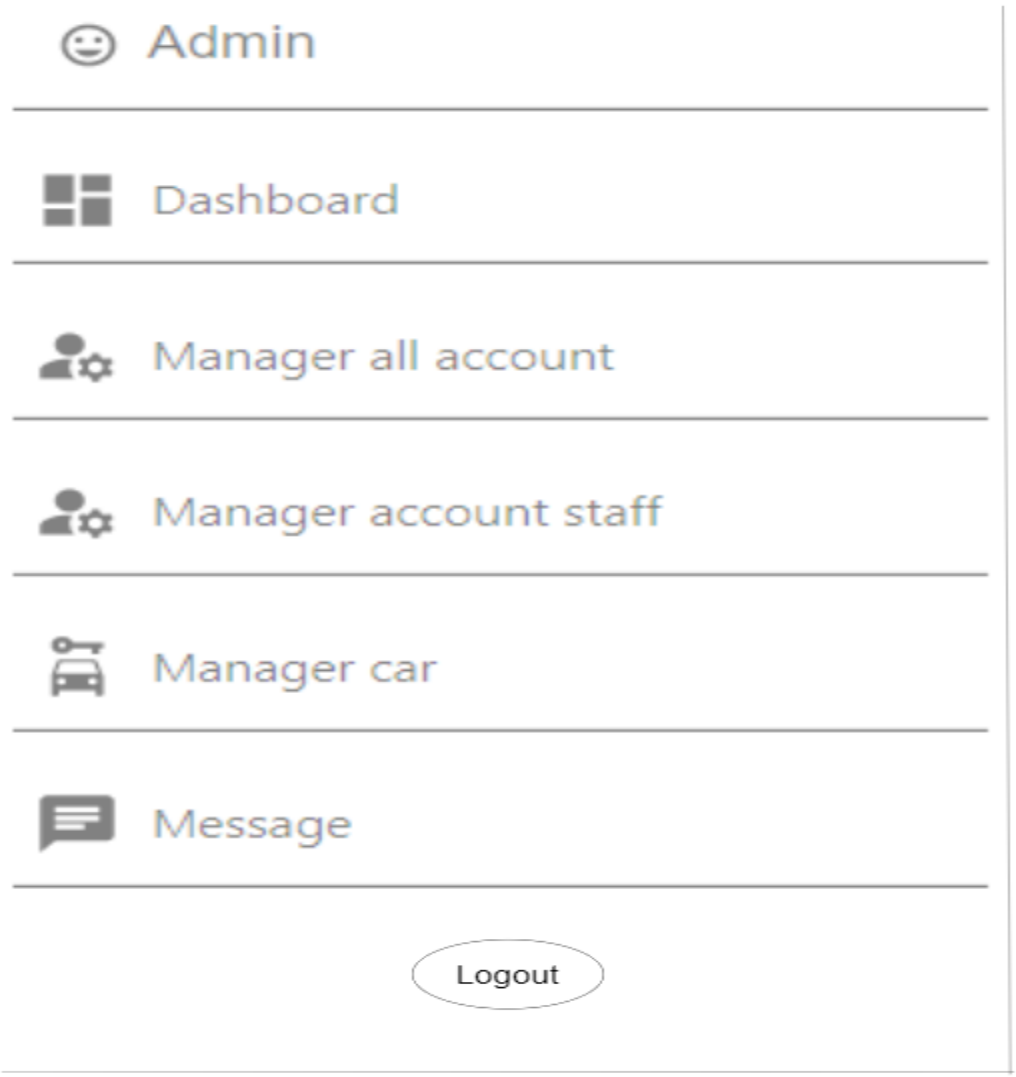



Figure 13: Navigation bar wireframe for admin.

This is the navigation bar wireframe when the user is login with staff role.

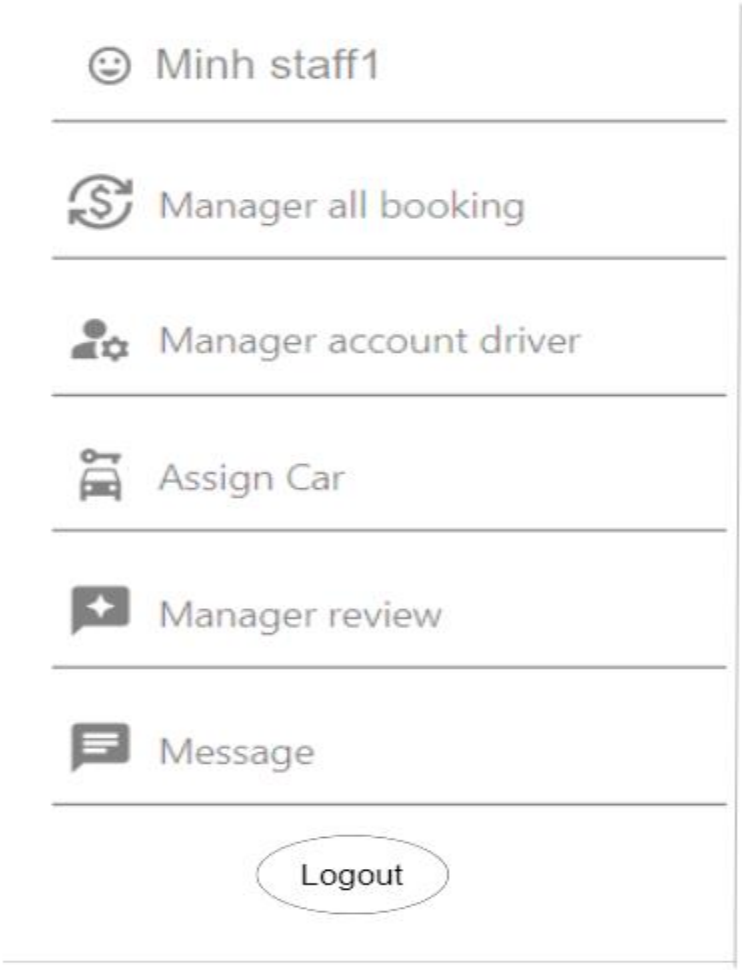



Figure 14: Navigation bar wireframe for staff.

This is the navigation bar wireframe when the user is login with driver and user role.

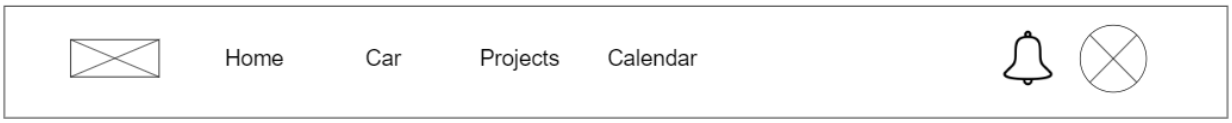



Figure 15: Navigation bar wireframe for driver and user.

### 8.6.2. Wireframes when user not yet login:

Register wireframe.

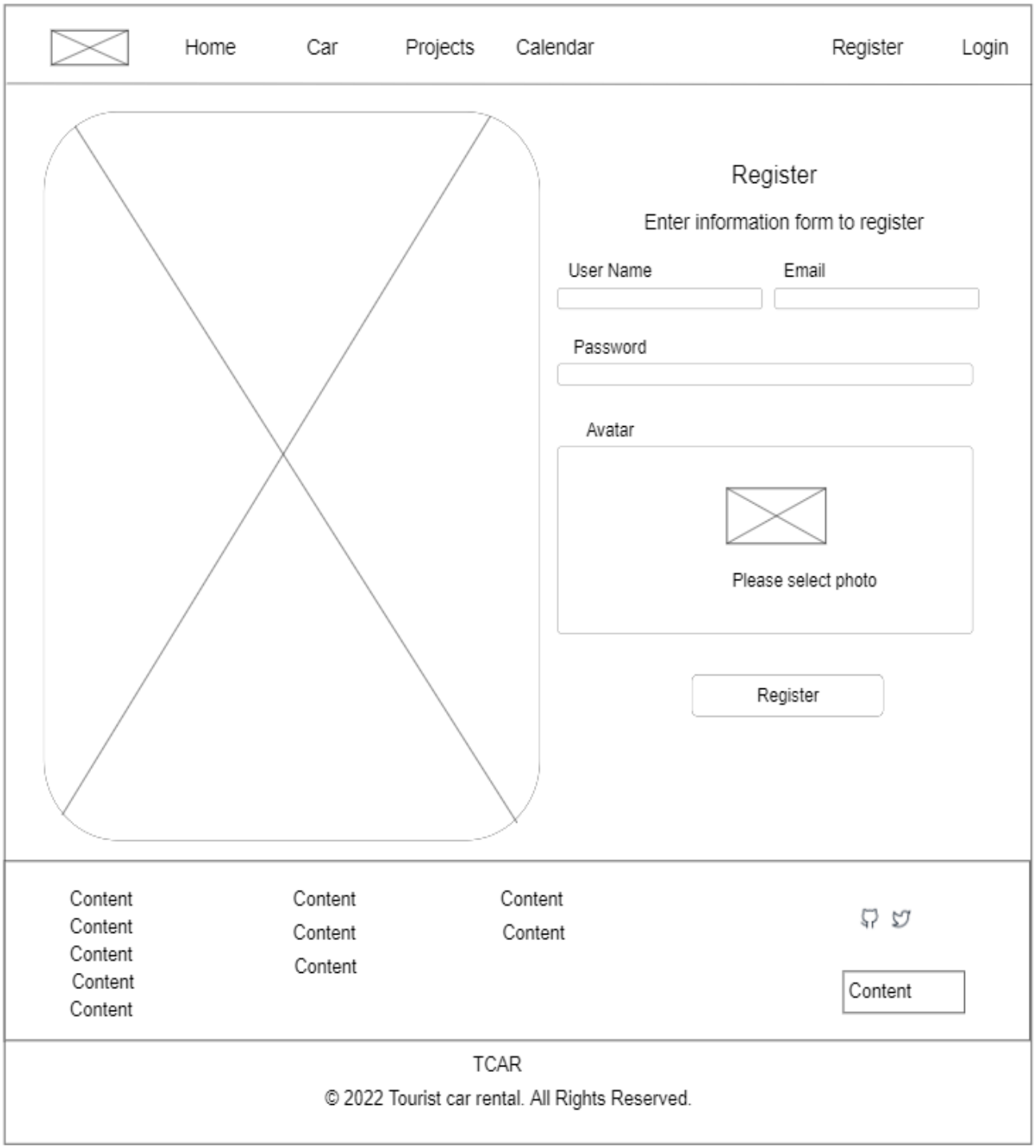



Figure 16: Register wireframe.

Login wireframe.

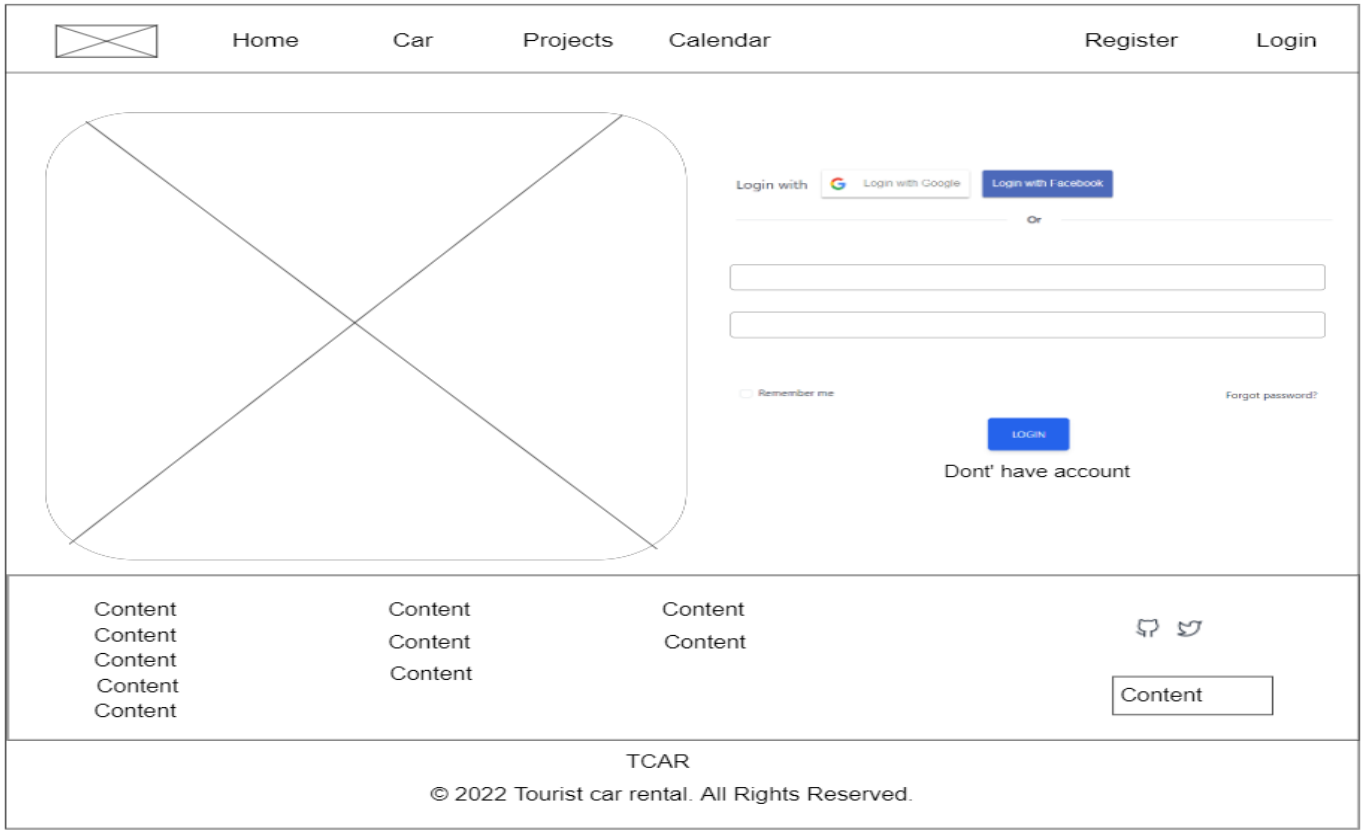



Figure 17: Login wireframe.

Forgot password wireframe.

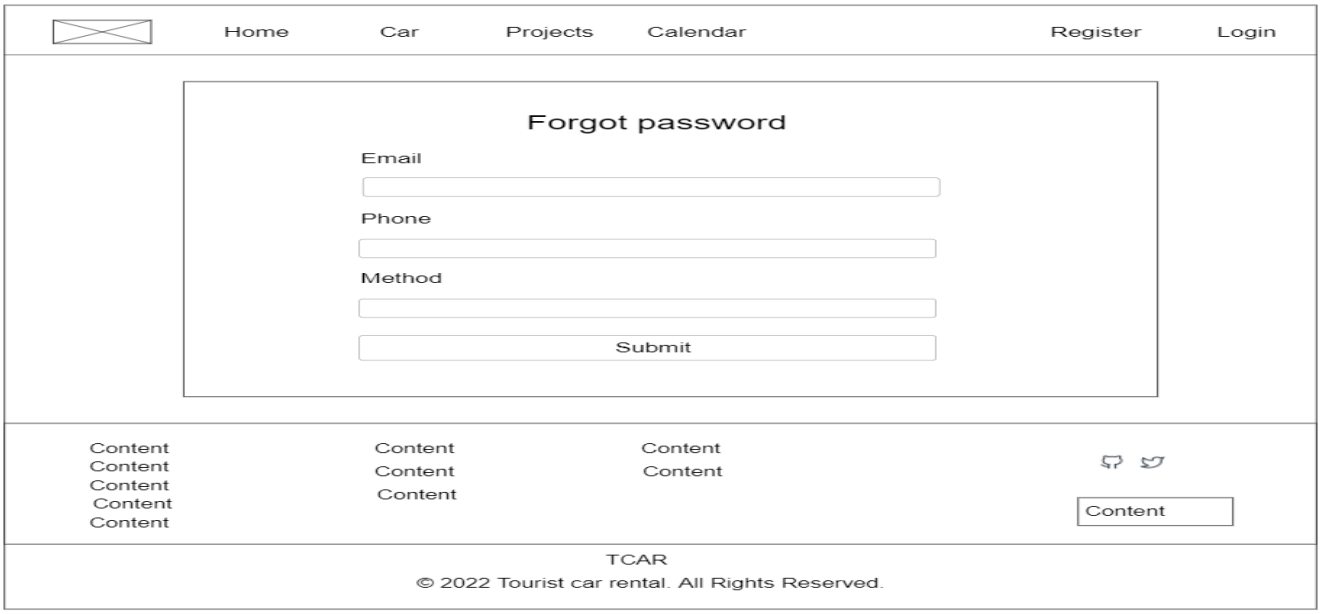



Figure 18: Forgot password wireframe.

Confirm OTP wireframe.

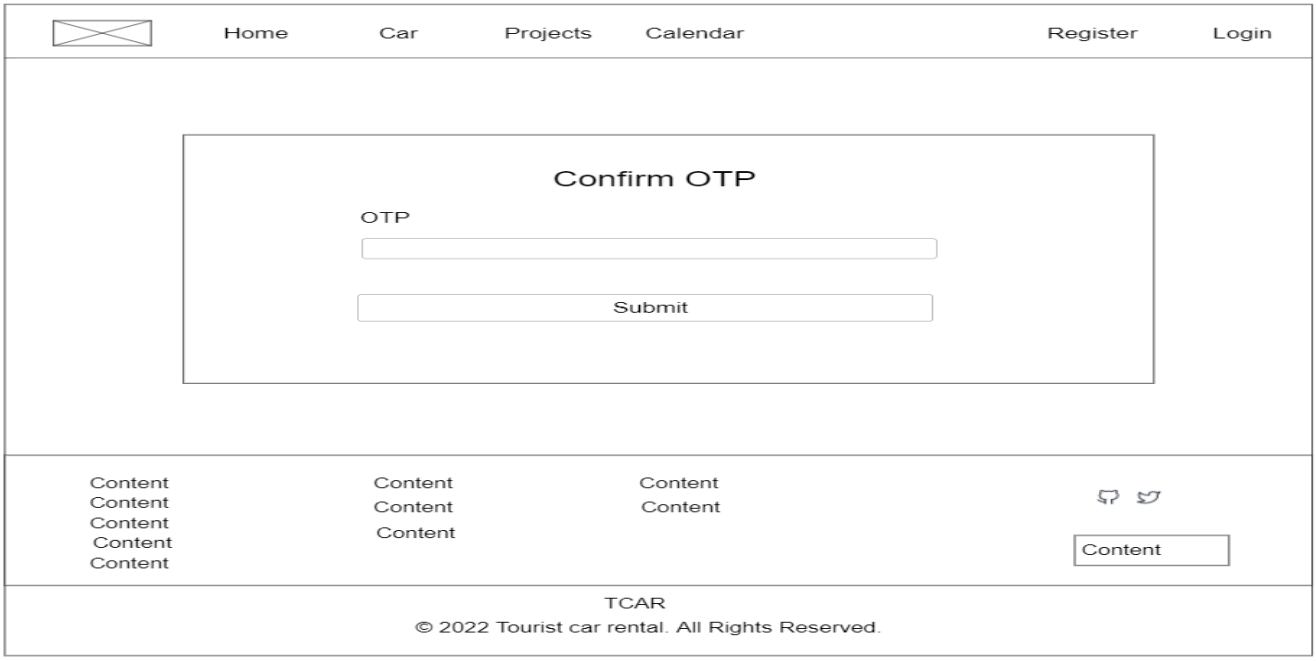



Figure 19: Confirm OTP wireframe.

Reset password wireframe.

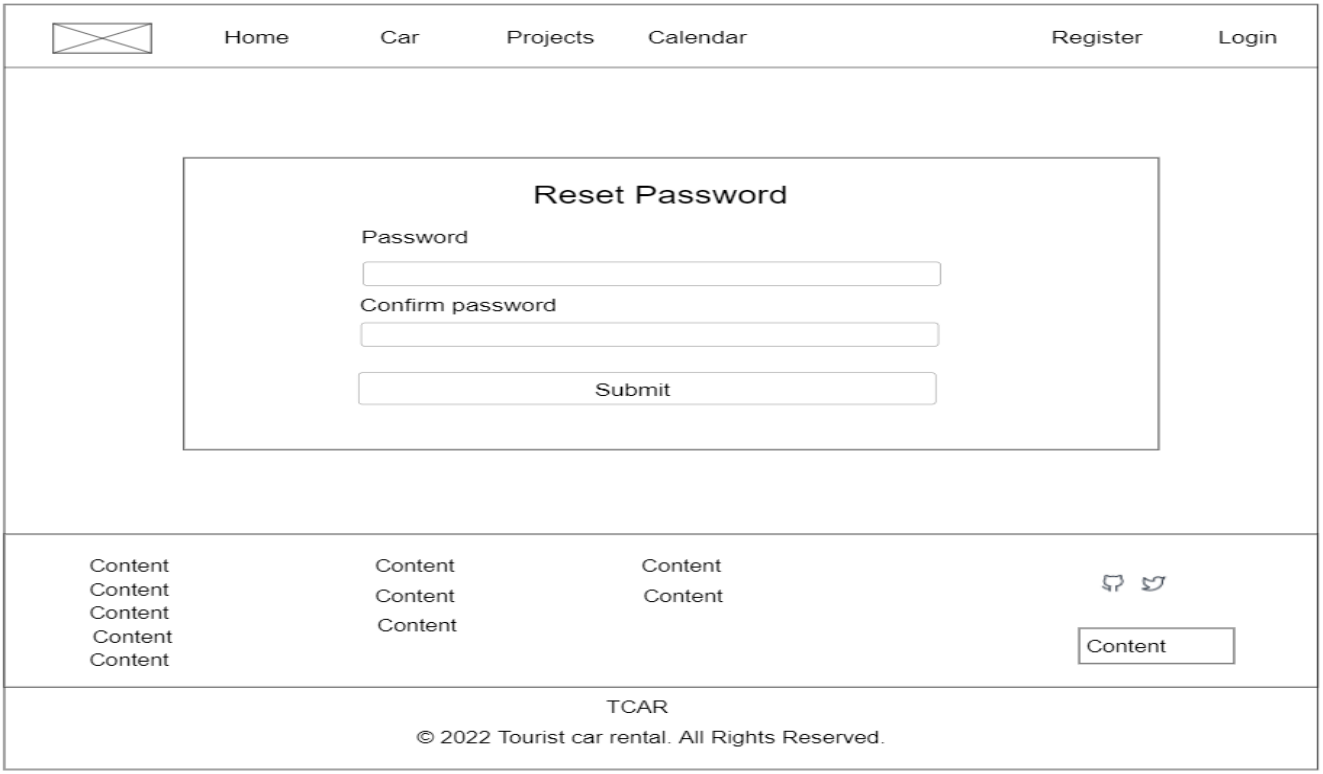



Figure 20: Reset password wireframe.

Home page wireframe.

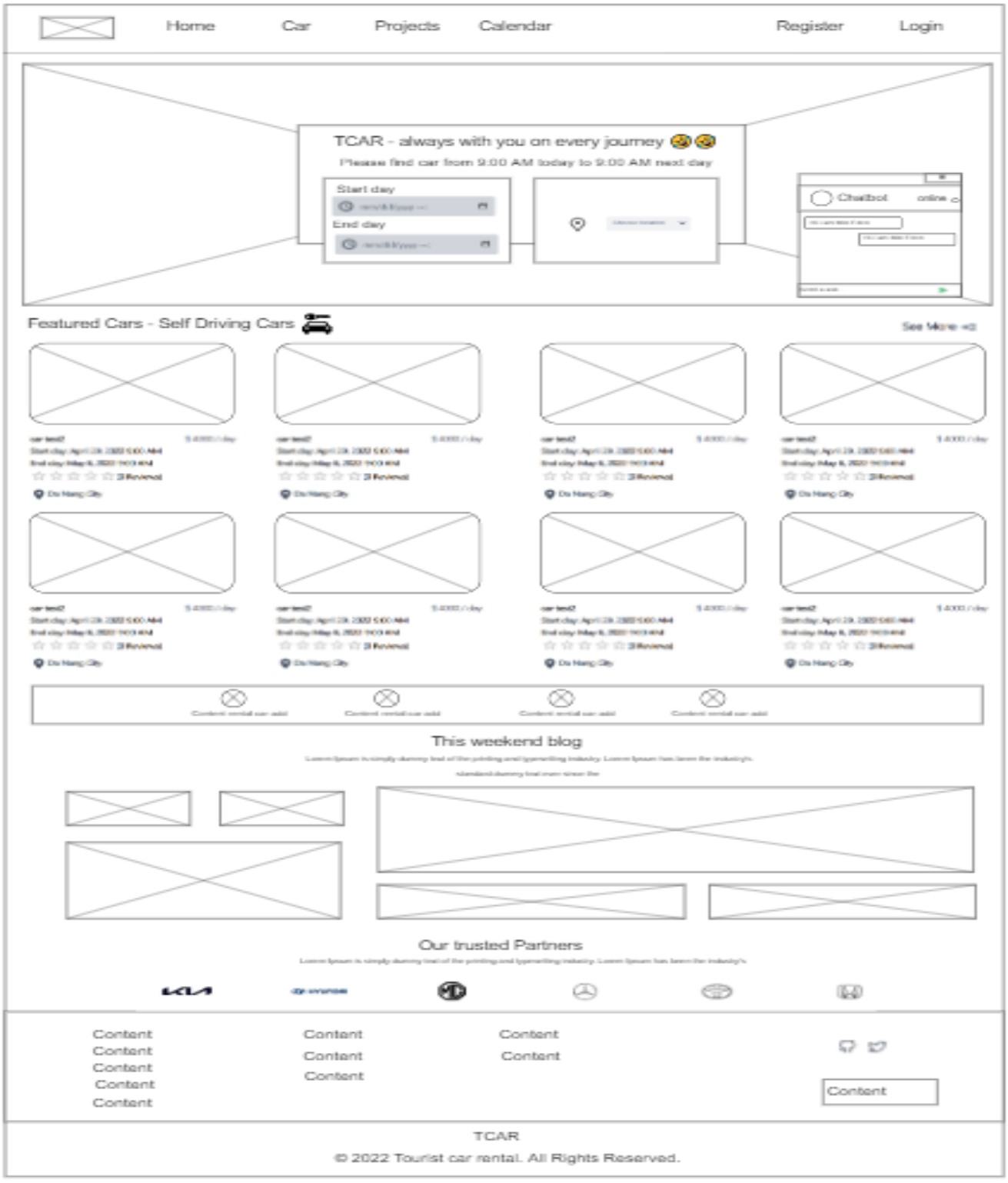



Figure 21: Home page wireframe.

### 8.6.3. Wireframes for admin:

Dashboard for admin wireframe.

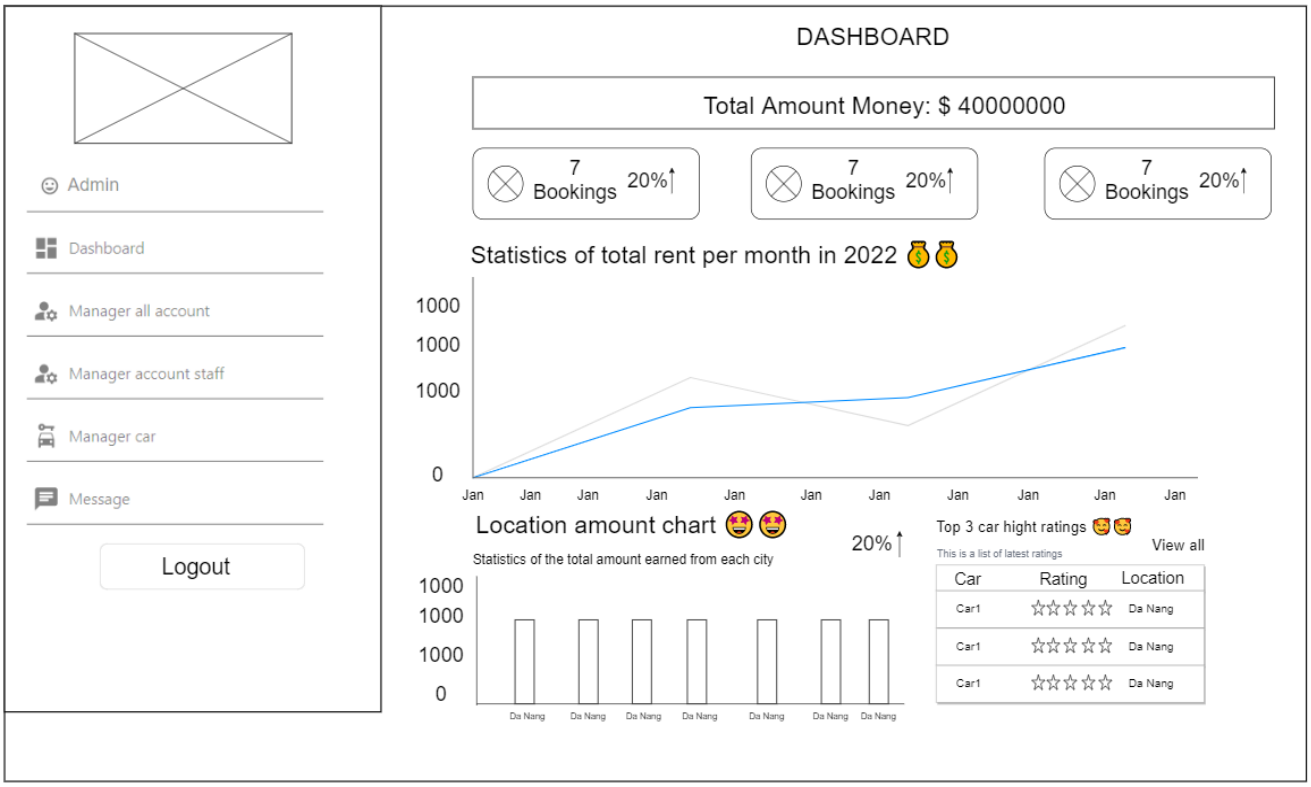



Figure 22: Dashboard wireframe.

Profile for admin wireframe.

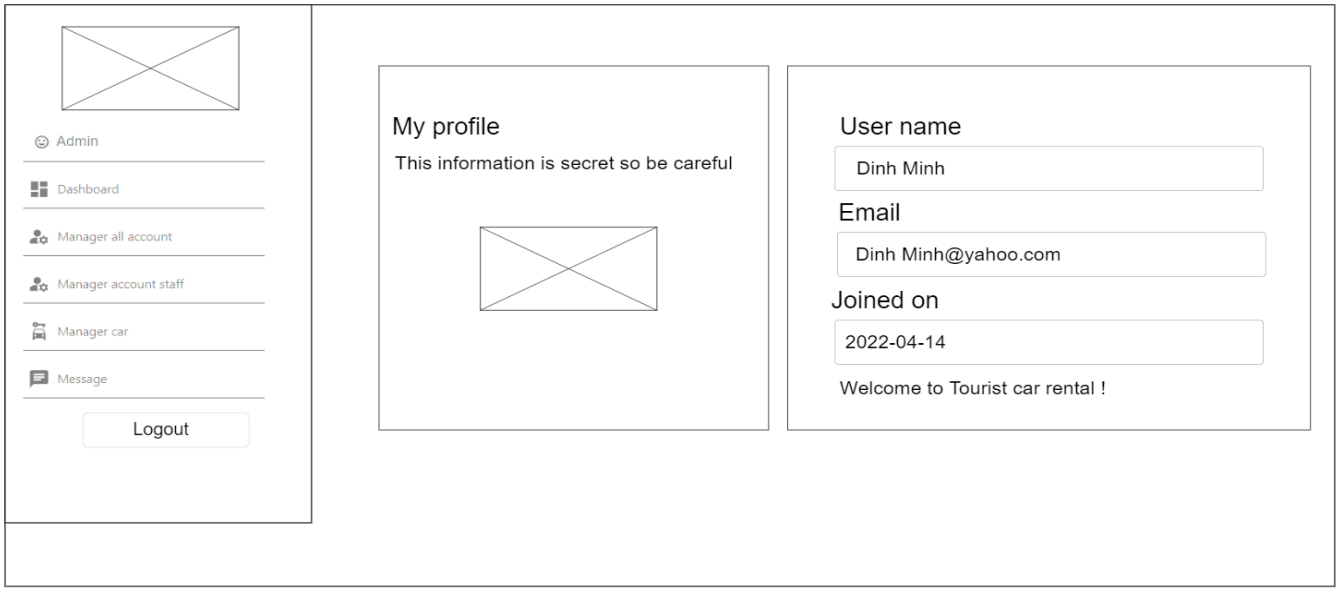



Figure 23: Profile for admin wireframe.

Edit profile for admin wireframe.

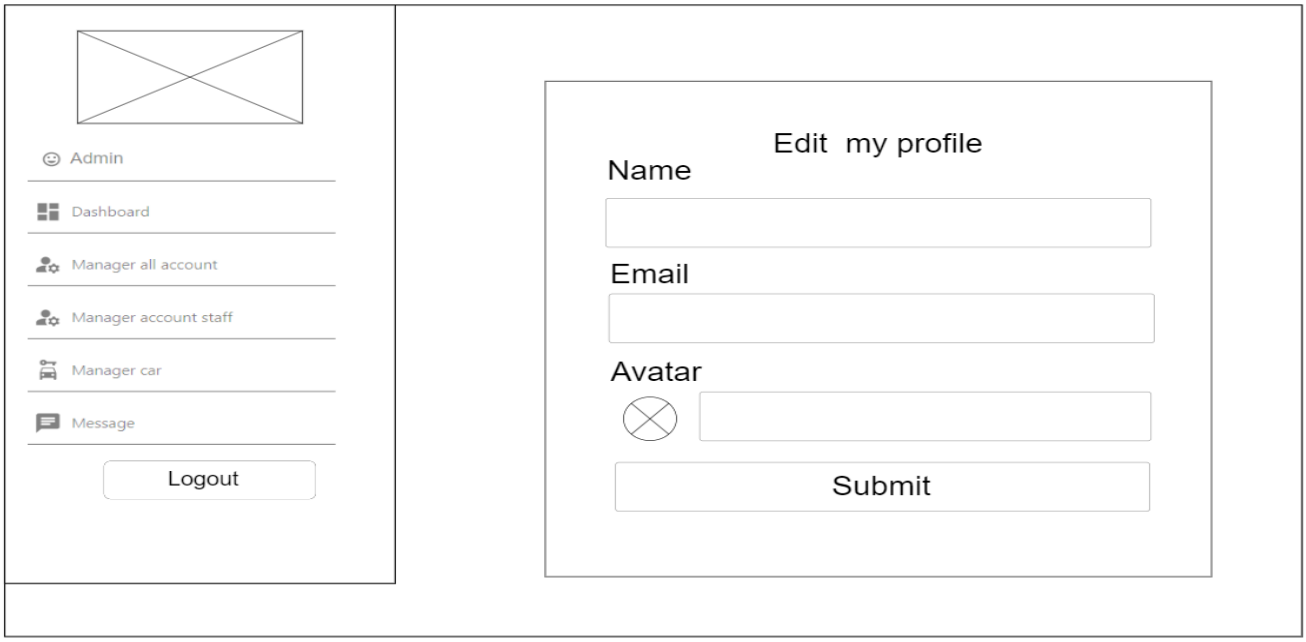



Figure 24: Edit profile for admin wireframe.

Change password for admin wireframe.

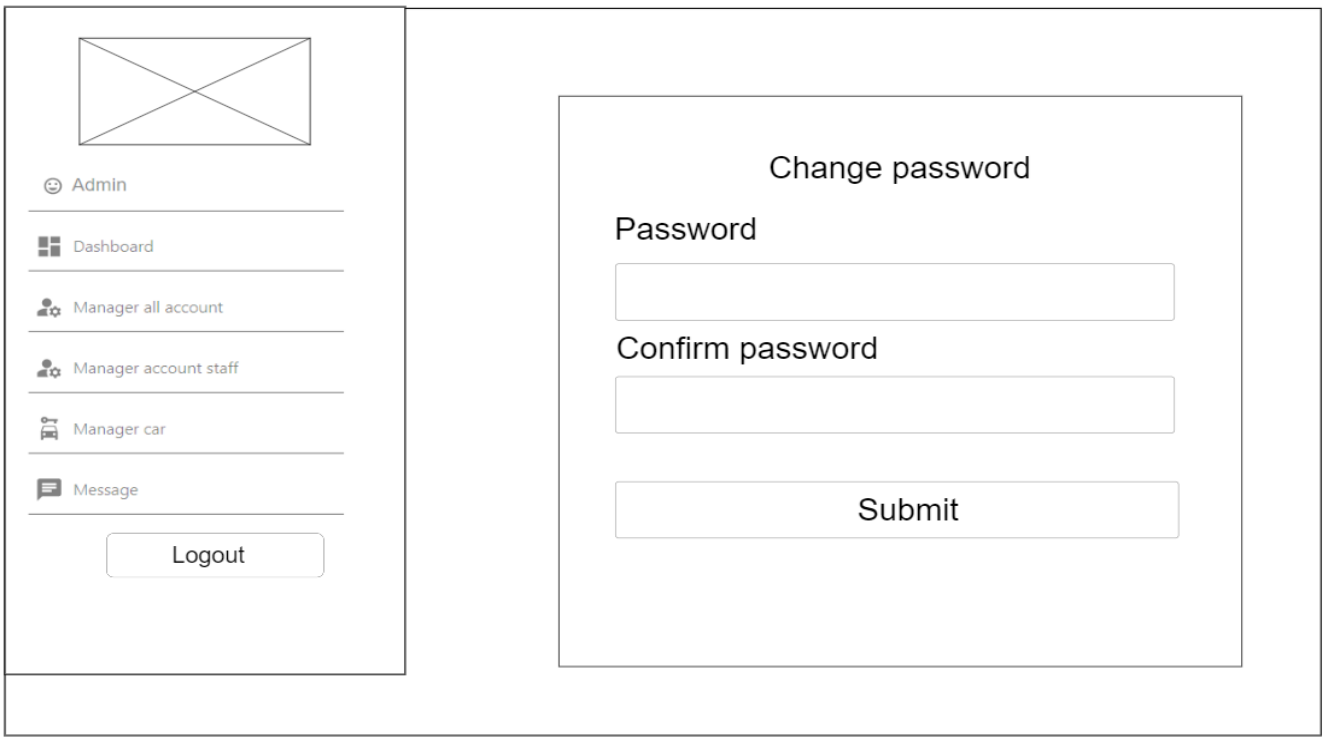



Figure 25: Change password for admin wireframe.

Manage all account for admin wireframe.

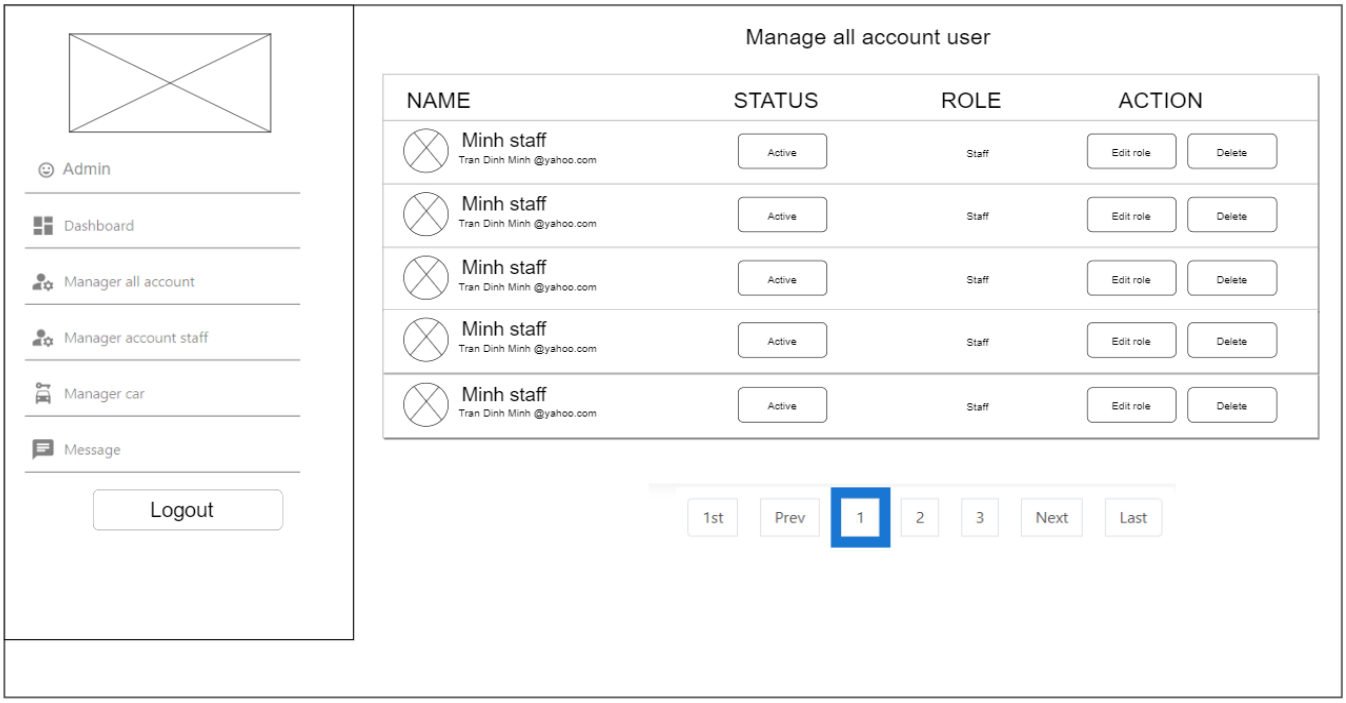



Figure 26: Manage all account for admin wireframe.

Edit role for admin wireframe.


Figure 27: Edit role for admin wireframe.

Manage all account staff for admin wireframe.

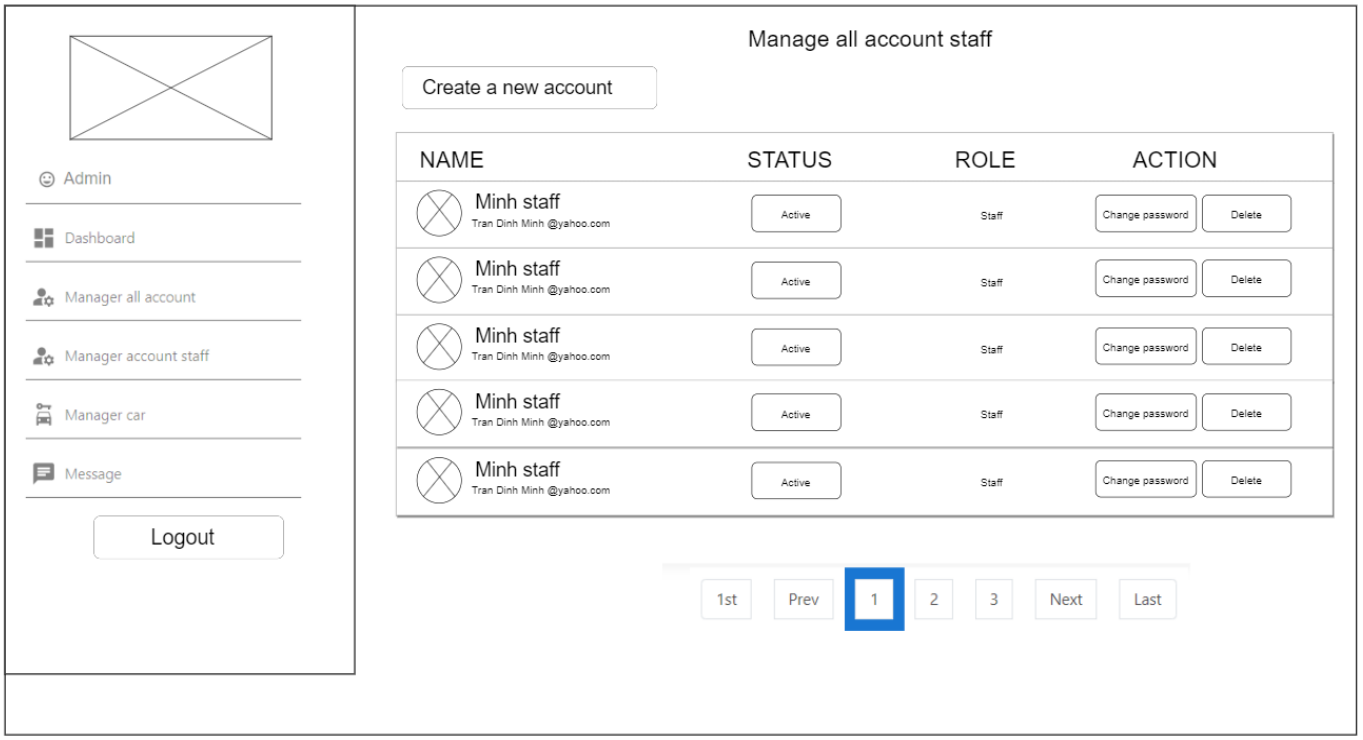



Figure 28: Manage all account staff for admin wireframe.

Create account staff for admin wireframe.

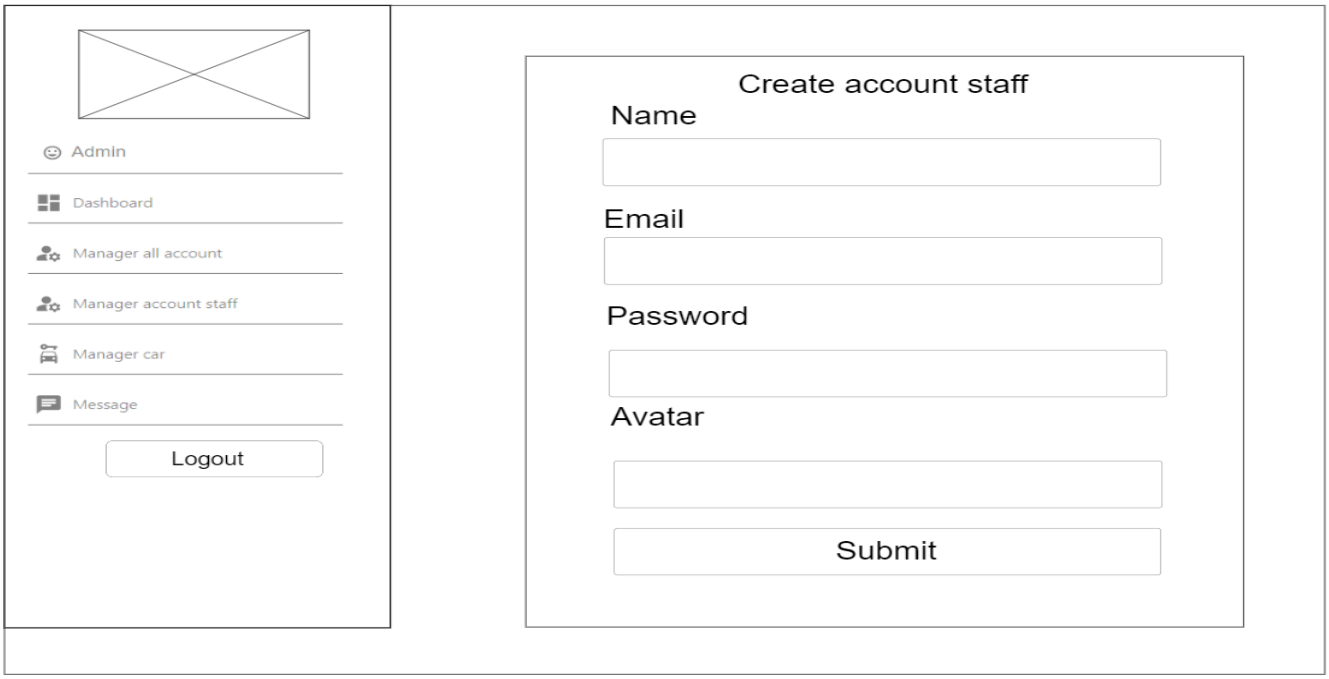



Figure 29: Create account staff for admin wireframe.

Change password staff for admin wireframe.

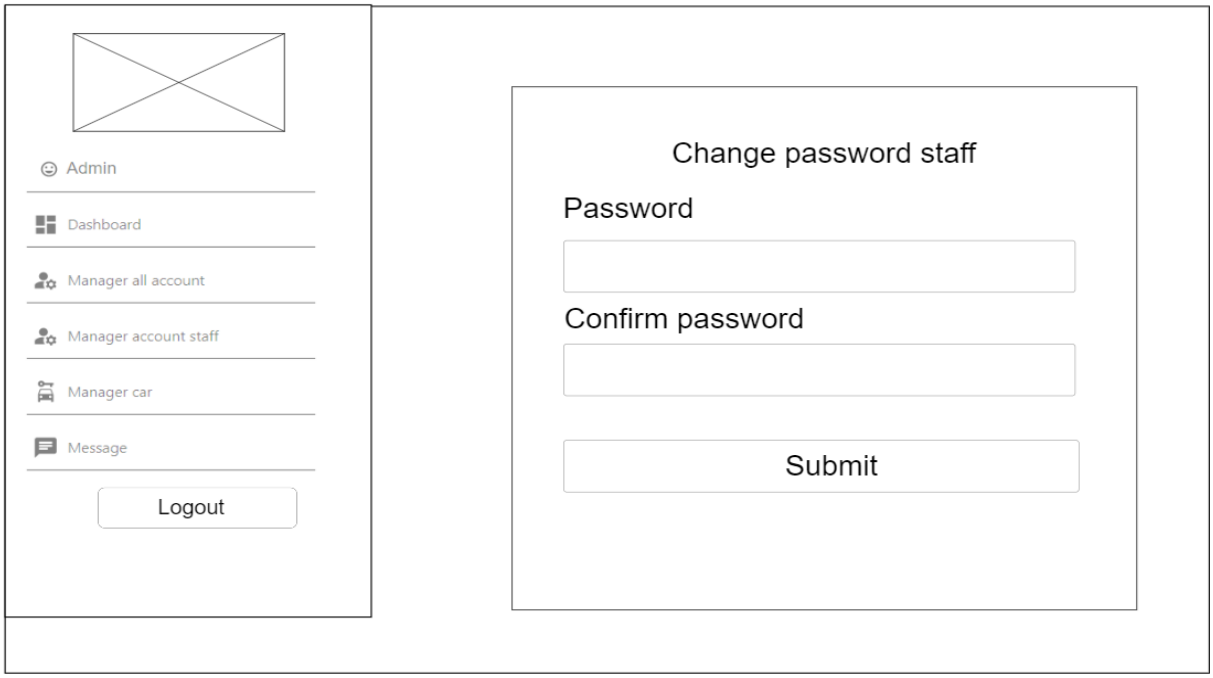



Figure 30: Change password staff for admin wireframe.

Manage all car for admin wireframe.

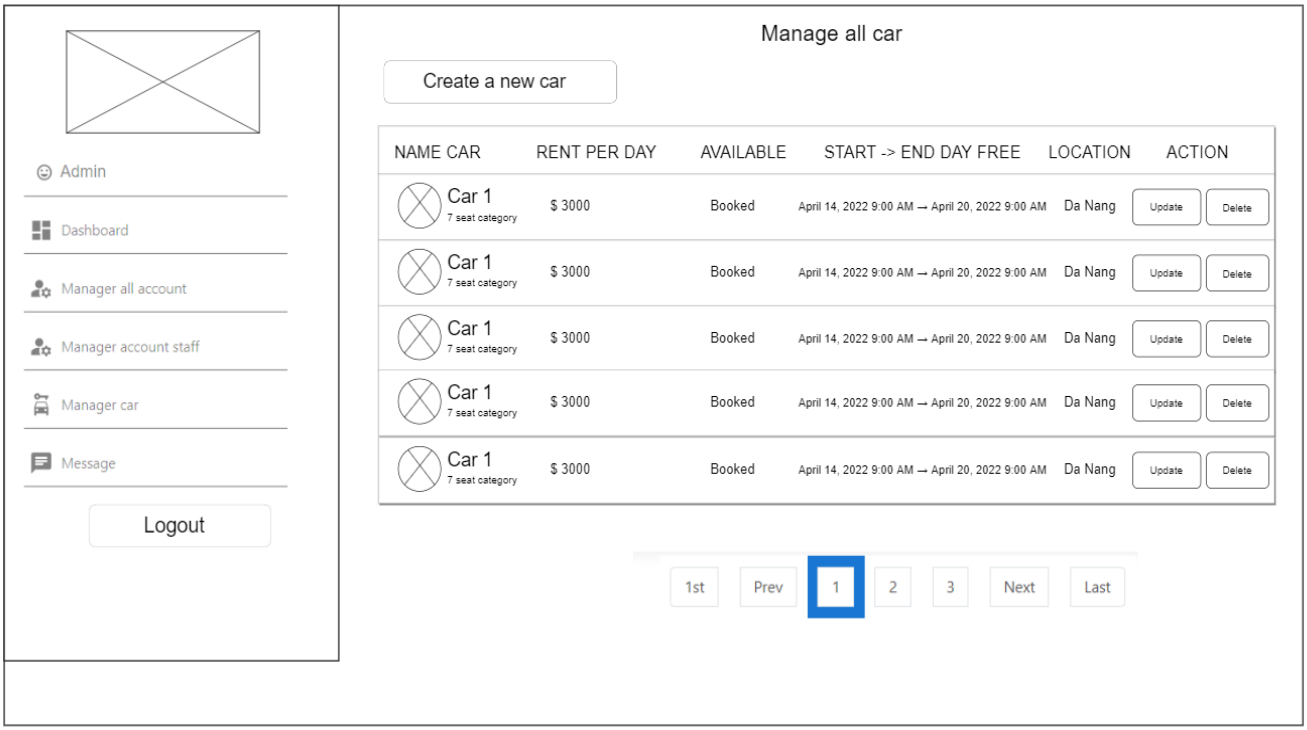



Figure 31: Manage all car for admin wireframe.

Create a new car for admin wireframe.

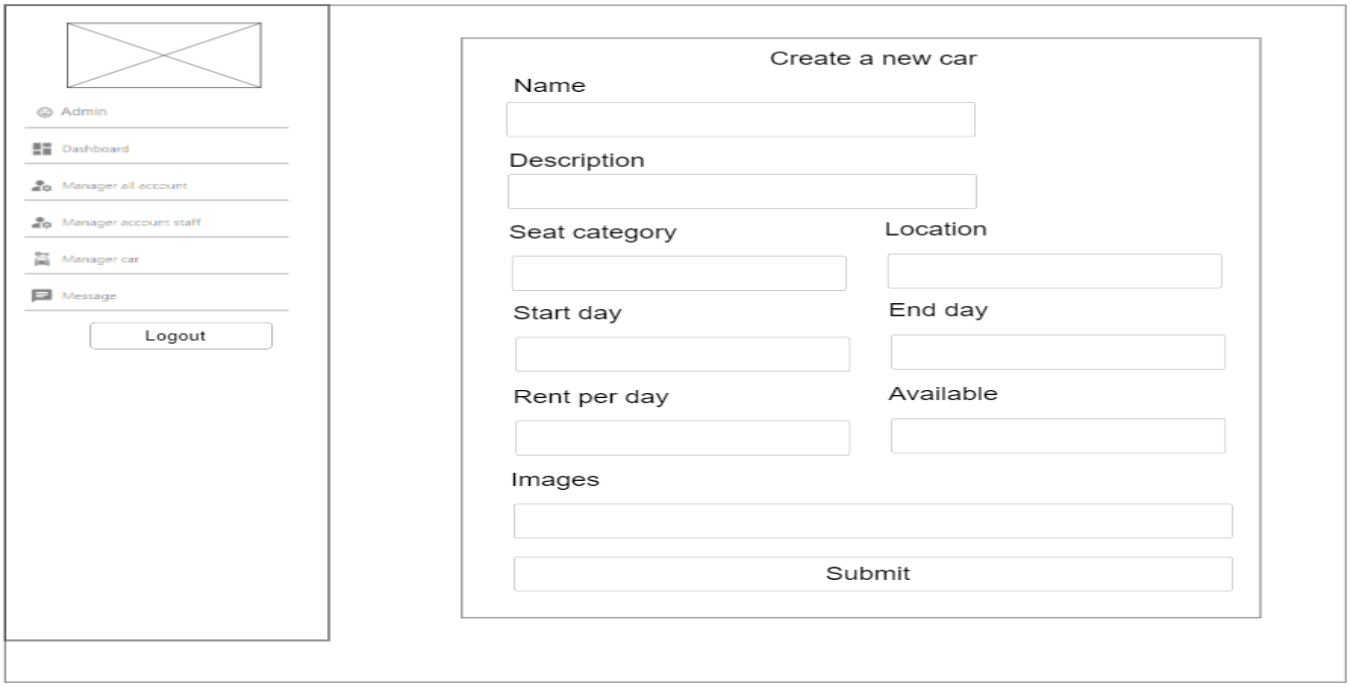



Figure 32: Create a new car for admin wireframe.

Update car for admin wireframe.

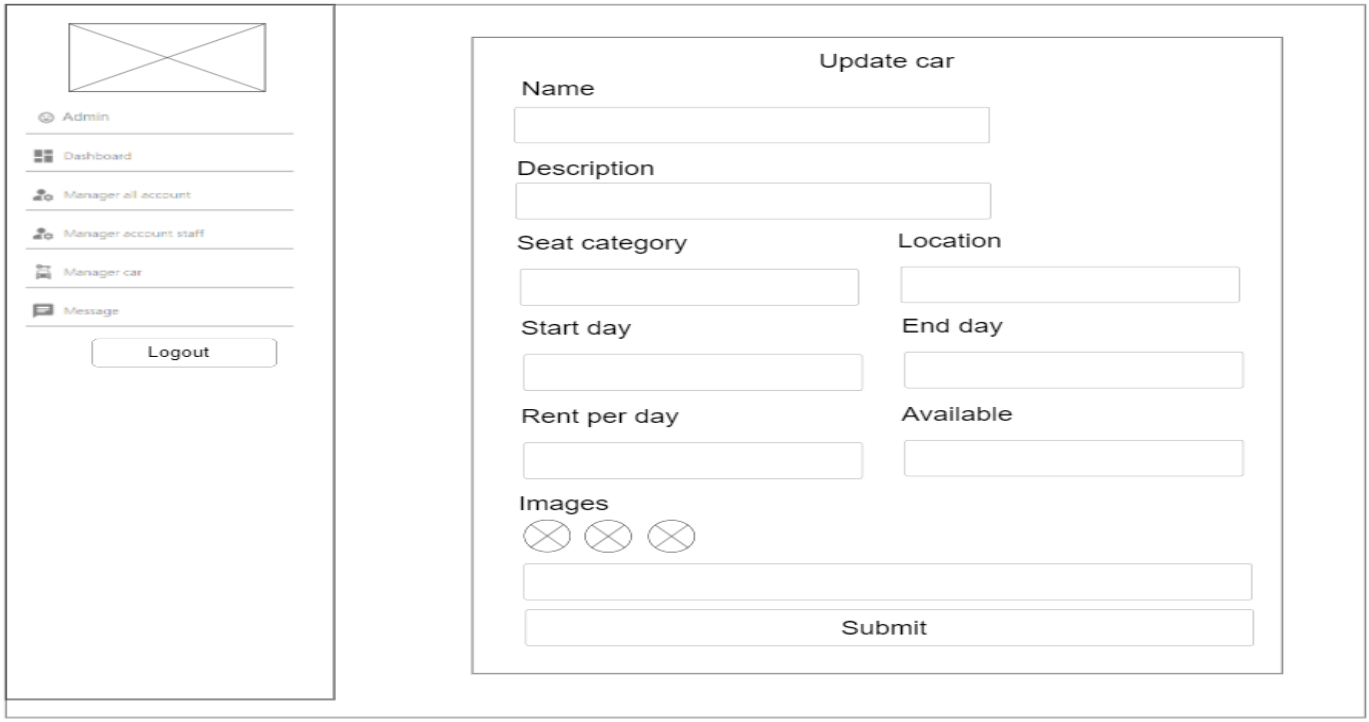



Figure 33: Update car for admin wireframe.

Chat message for admin wireframe.

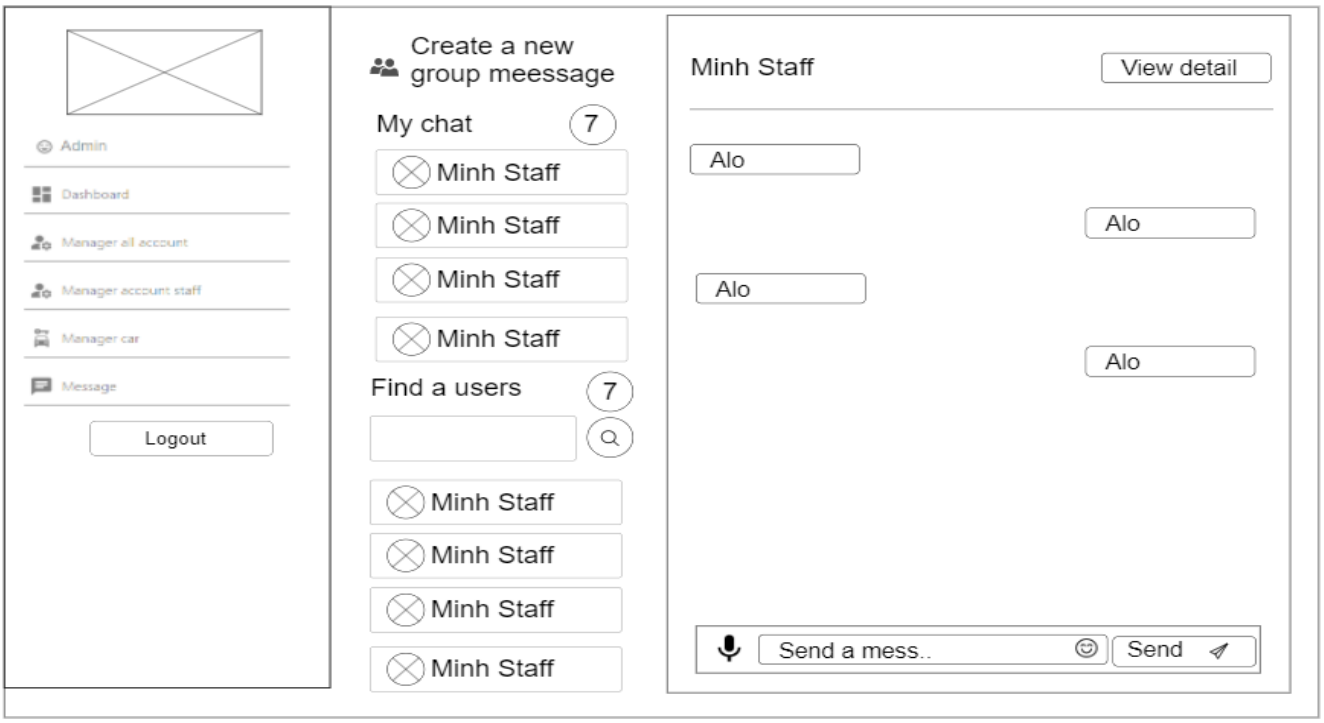



Figure 34: Chat message for admin wireframe.

See detail user chat for admin wireframe.

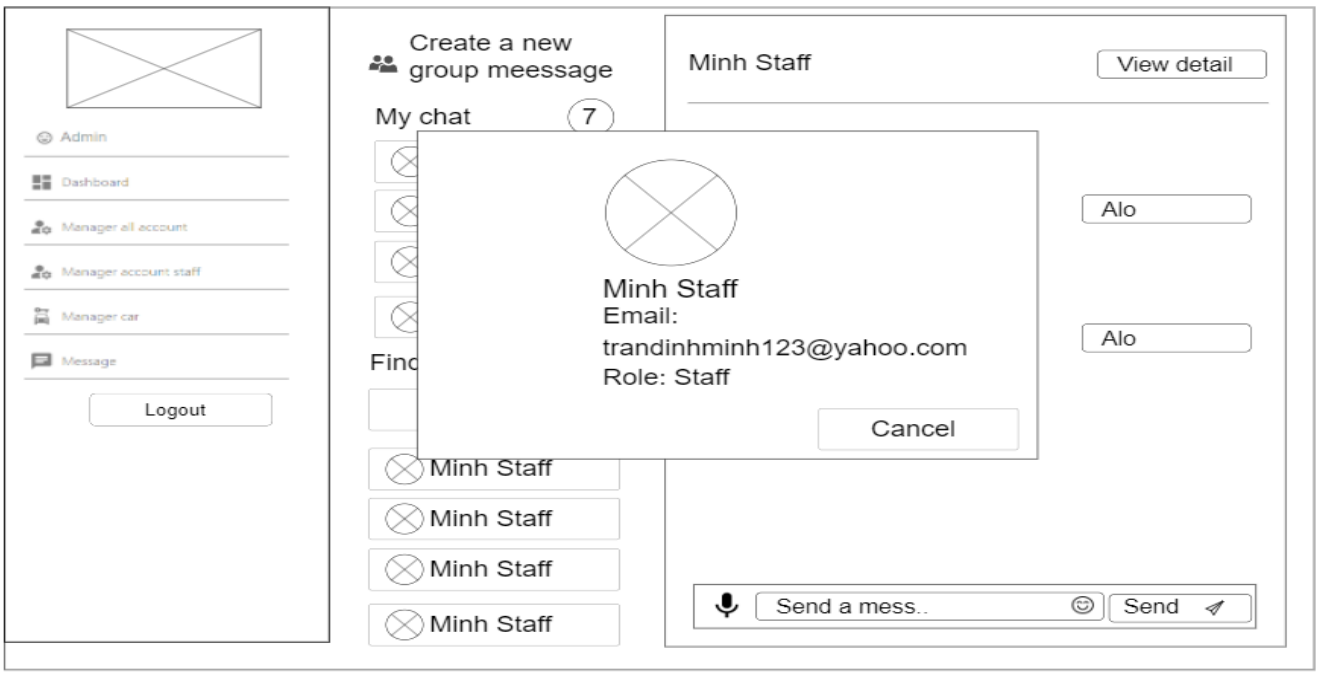



Figure 35: View detail user chat for admin wireframe.

Create a new group chat for admin wireframe.

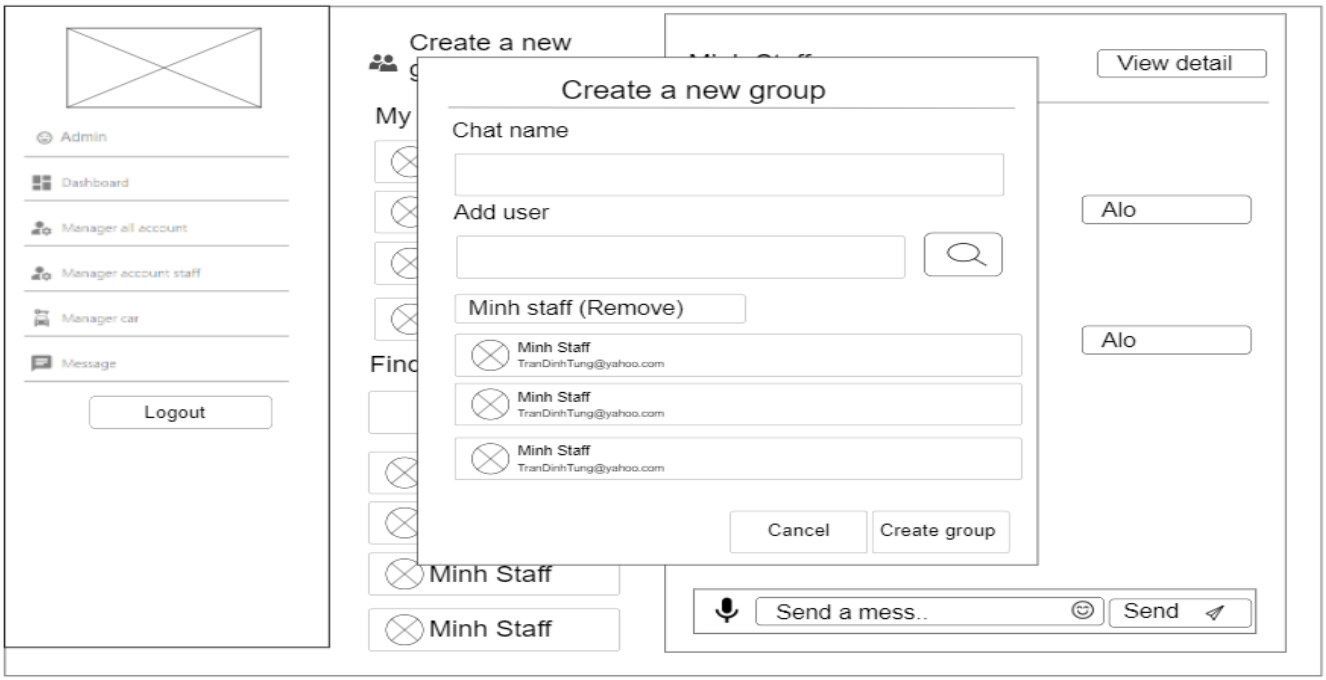



Figure 36: Create a new group chat for admin wireframe.

Update name, remove user, leave group for admin wireframe.

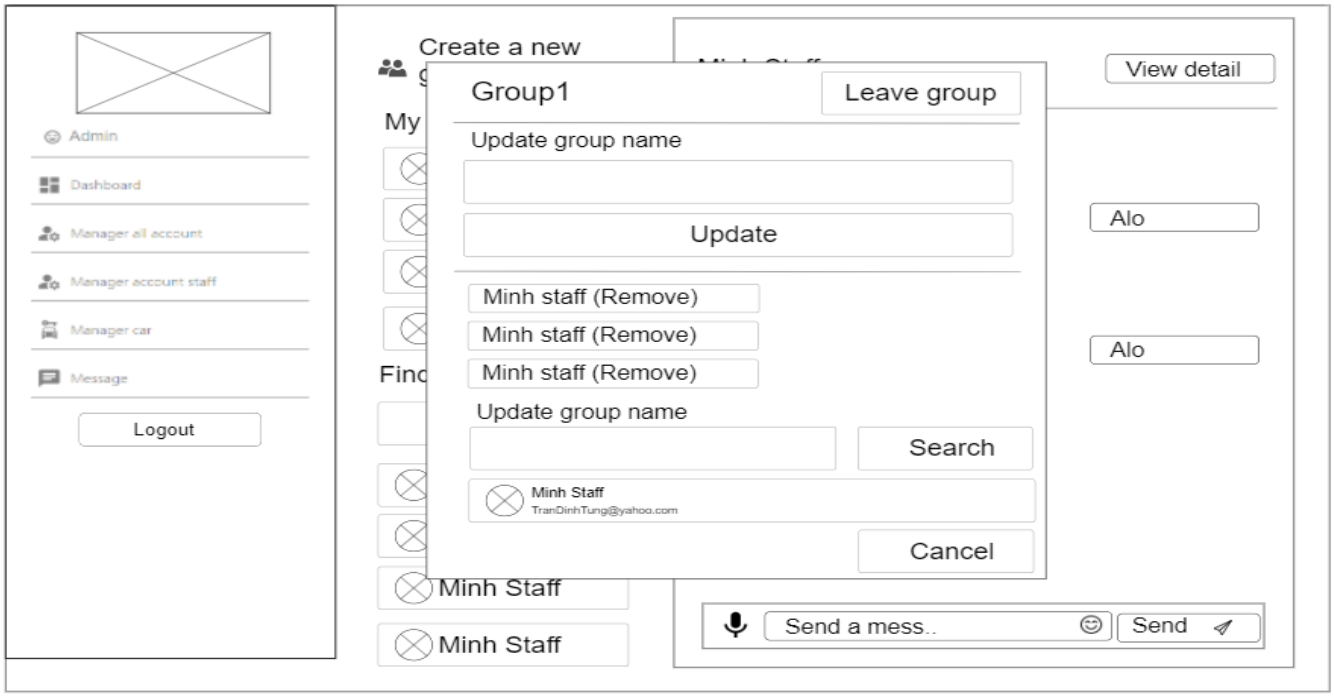



Figure 37: Update name, remove user, leave group for admin wireframe.

### 8.6.4. Wireframes for staff:

Manage all booking for staff wireframe.

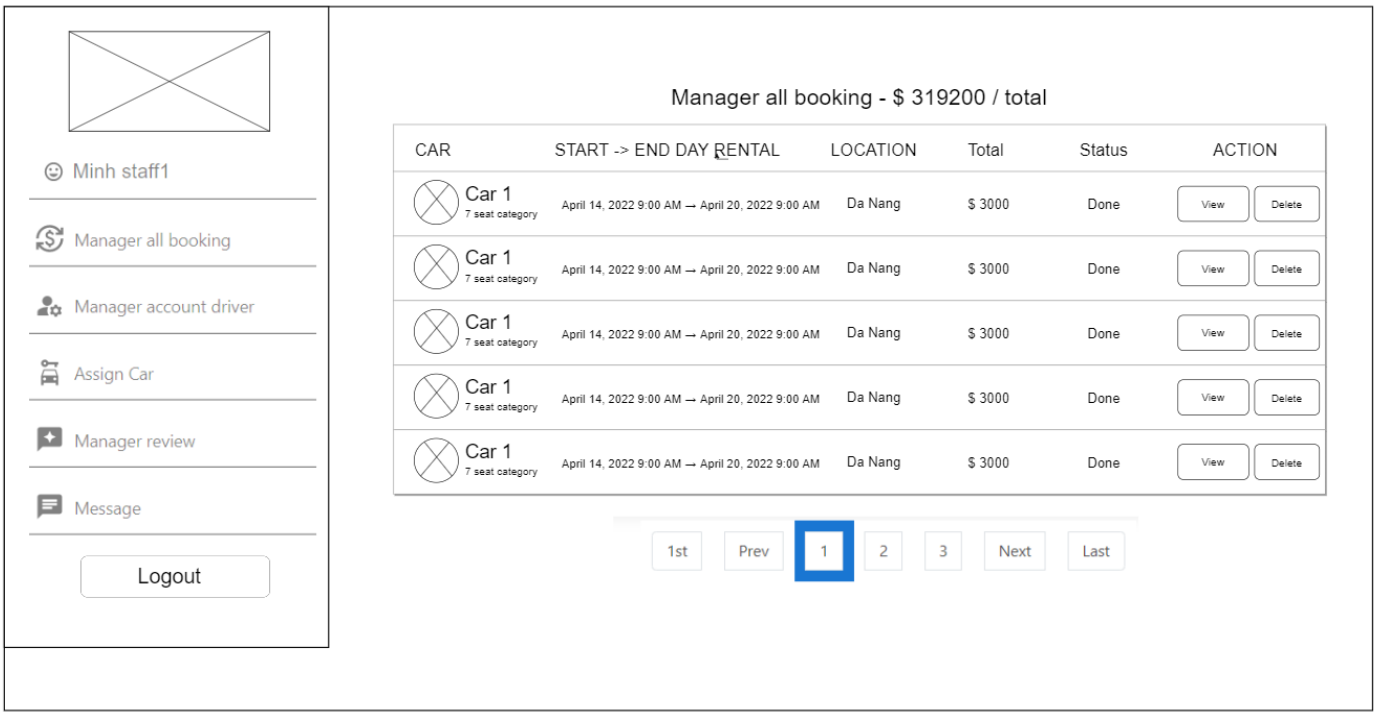



Figure 38: Manage all booking for staff wireframe.

View details booking car of user for staff wireframe.

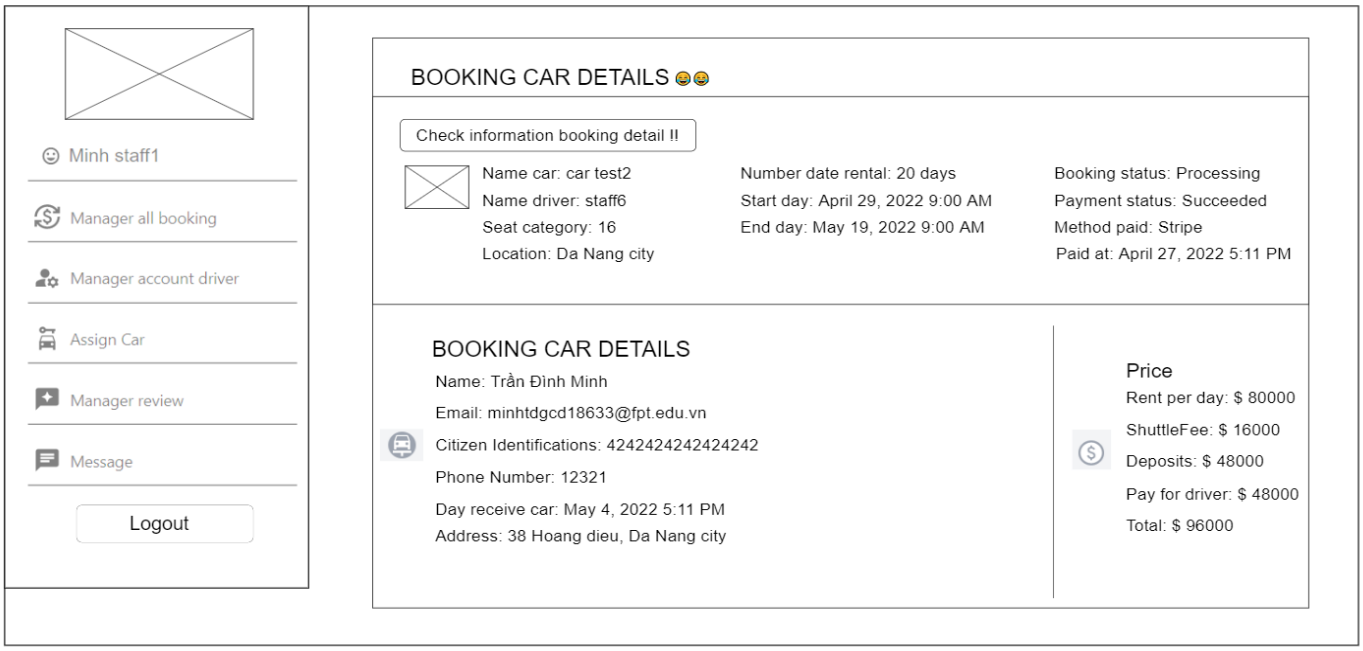



Figure 39: View details booking car of user for staff wireframe.

Manage all account driver for staff wireframes.

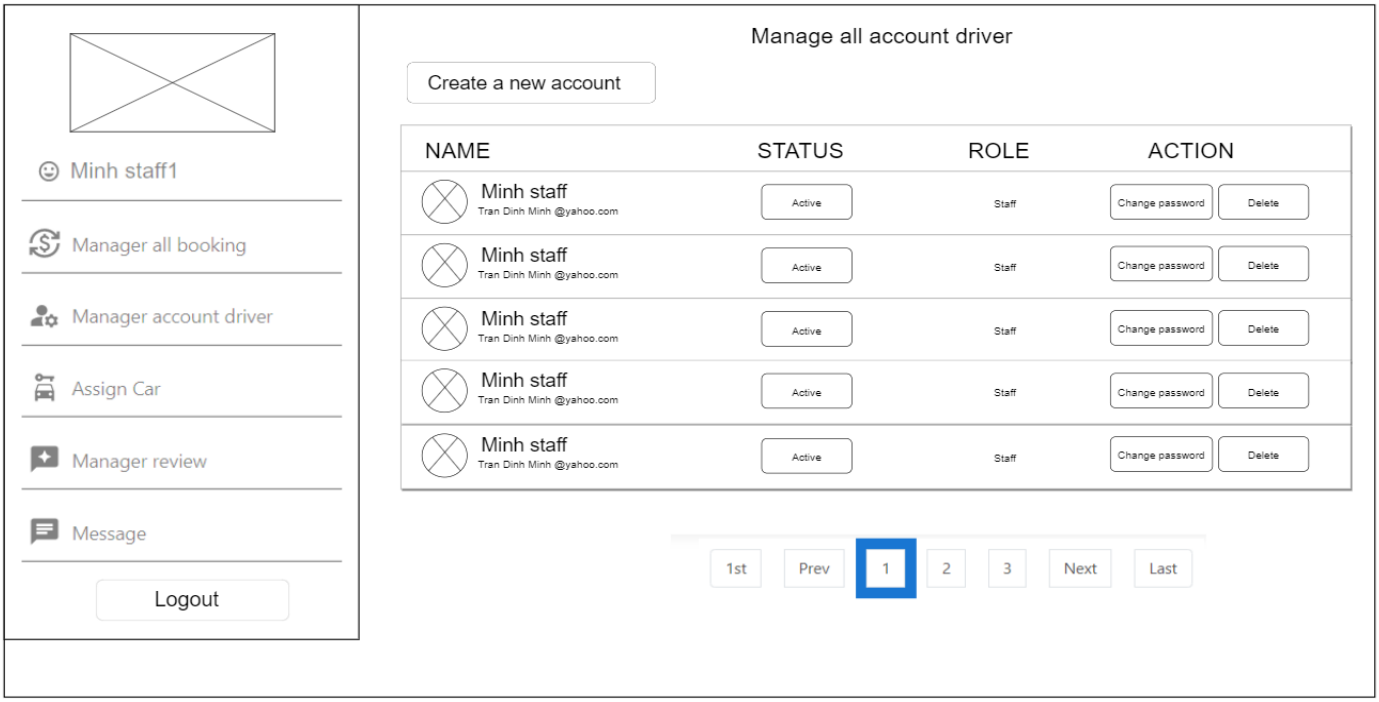



Figure 40: Manage all account driver for staff wireframes.

Create account driver for staff wireframe.

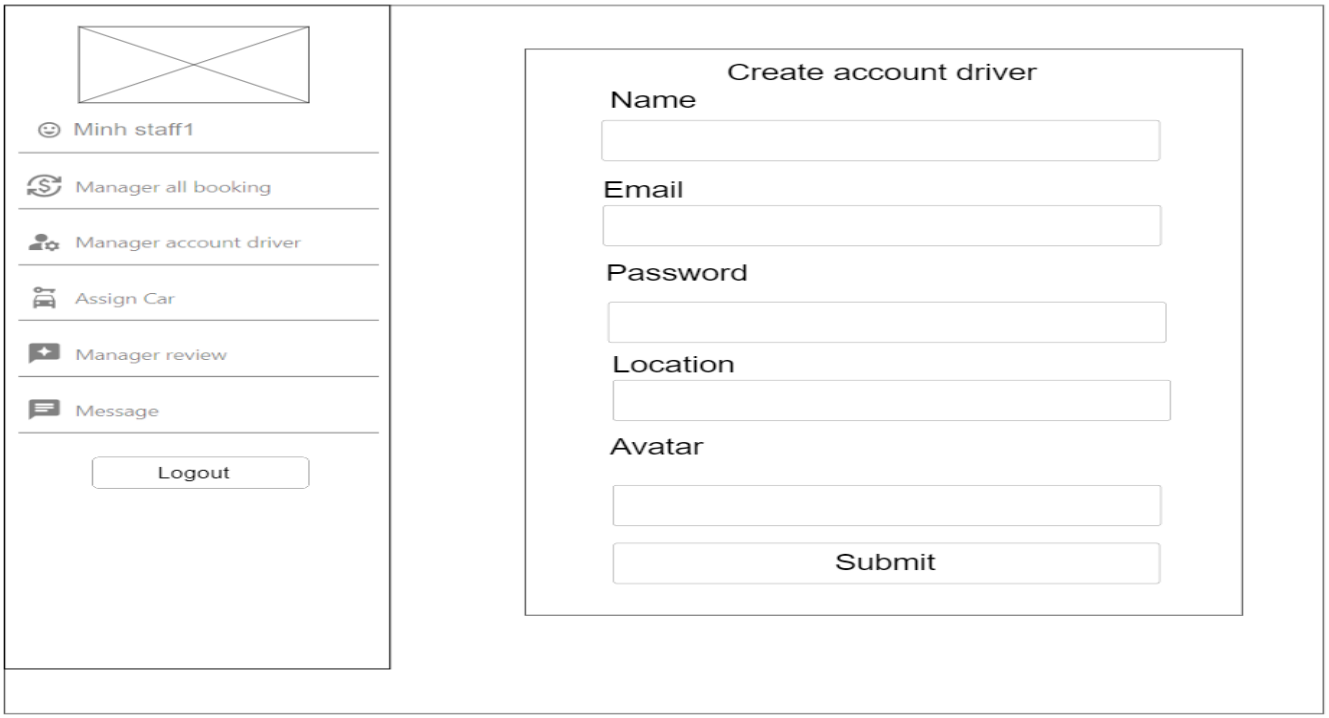



Figure 41: Create account driver for staff wireframe.

Change password driver for staff wireframe.

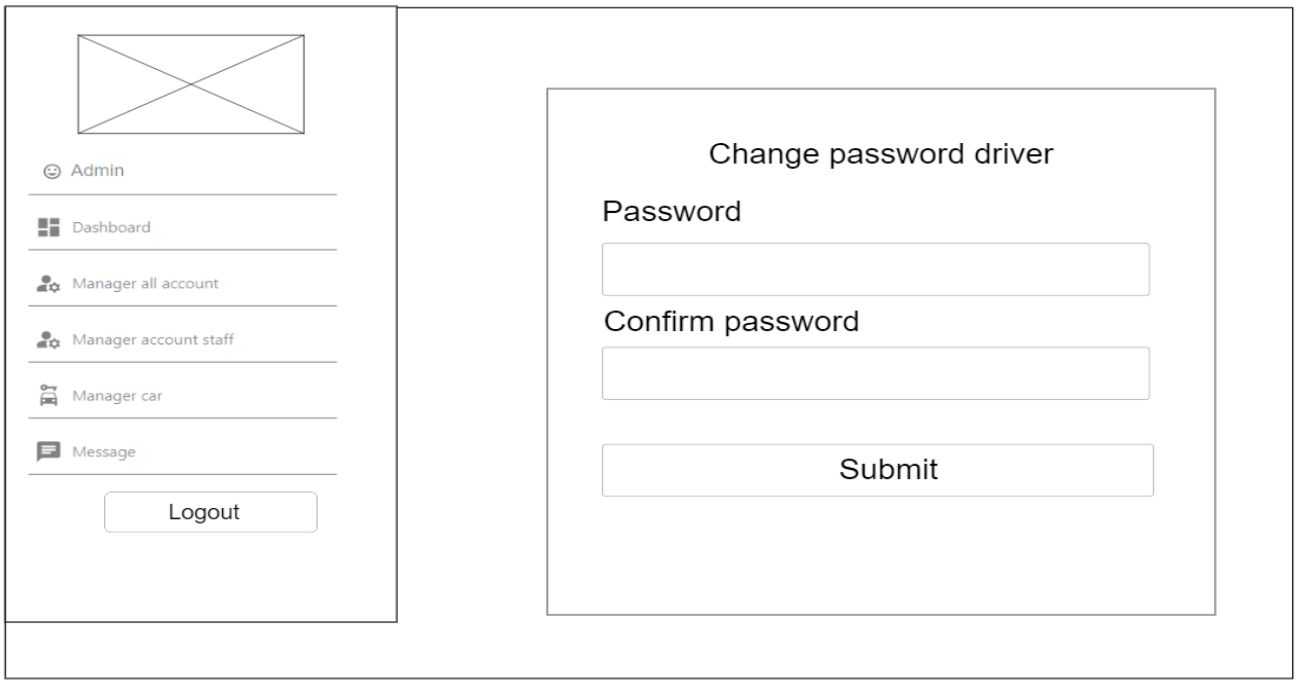



Figure 42: Change password driver for staff wireframe.

All cars do not assign and assign to driver for staff wireframe.

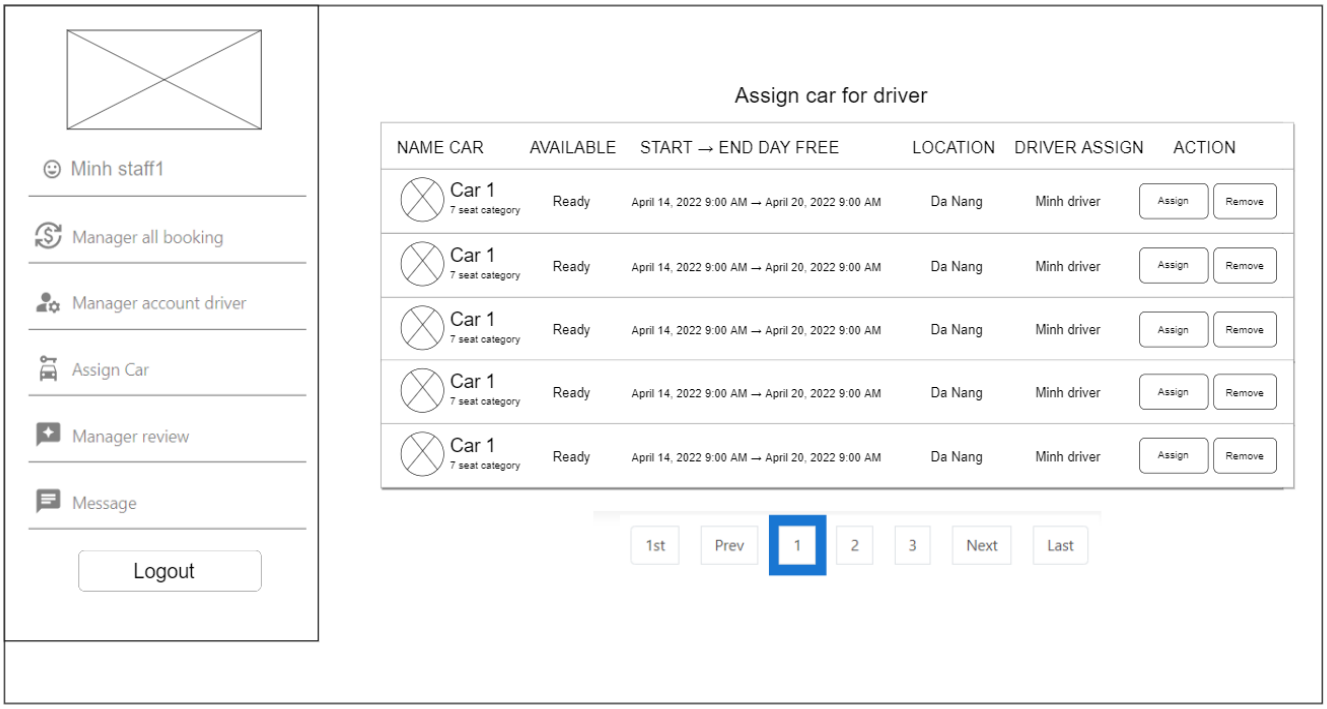



Figure 43: All cars do not assign and assign to driver for staff wireframe.

All list drivers do not assign car for staff wireframe.

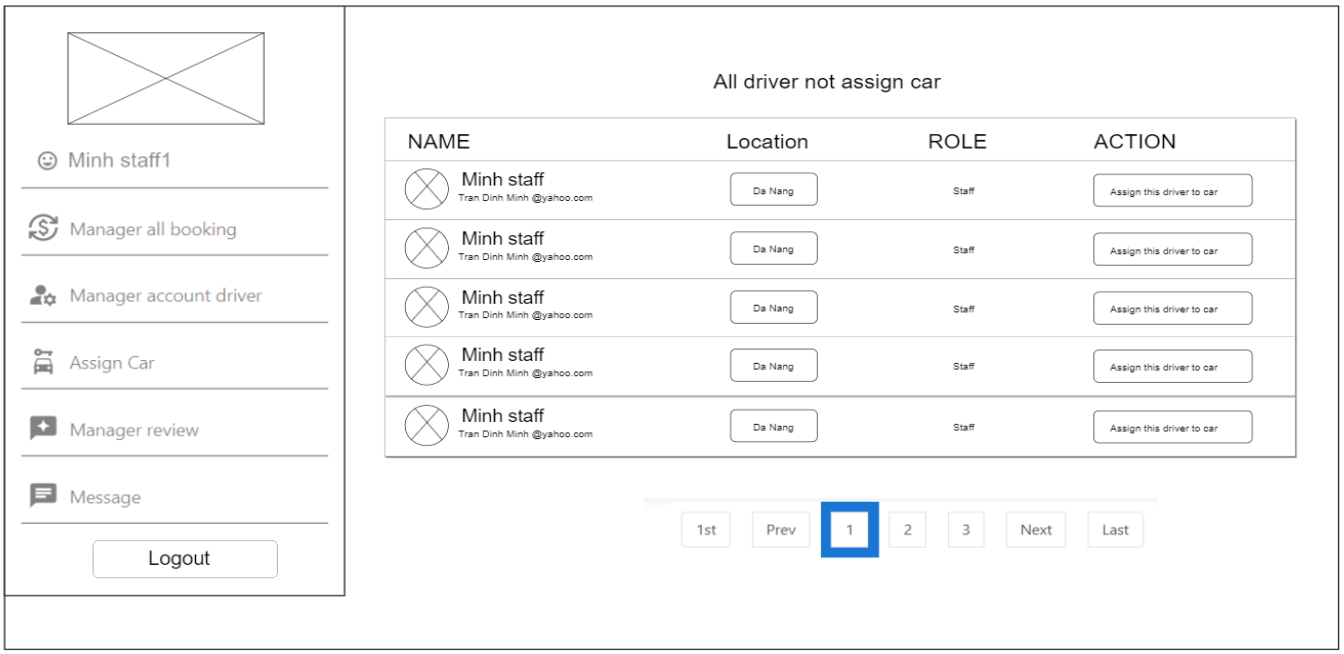



Figure 44: All list drivers dot not assign car for staff wireframe.

Manager all review for staff wireframe.

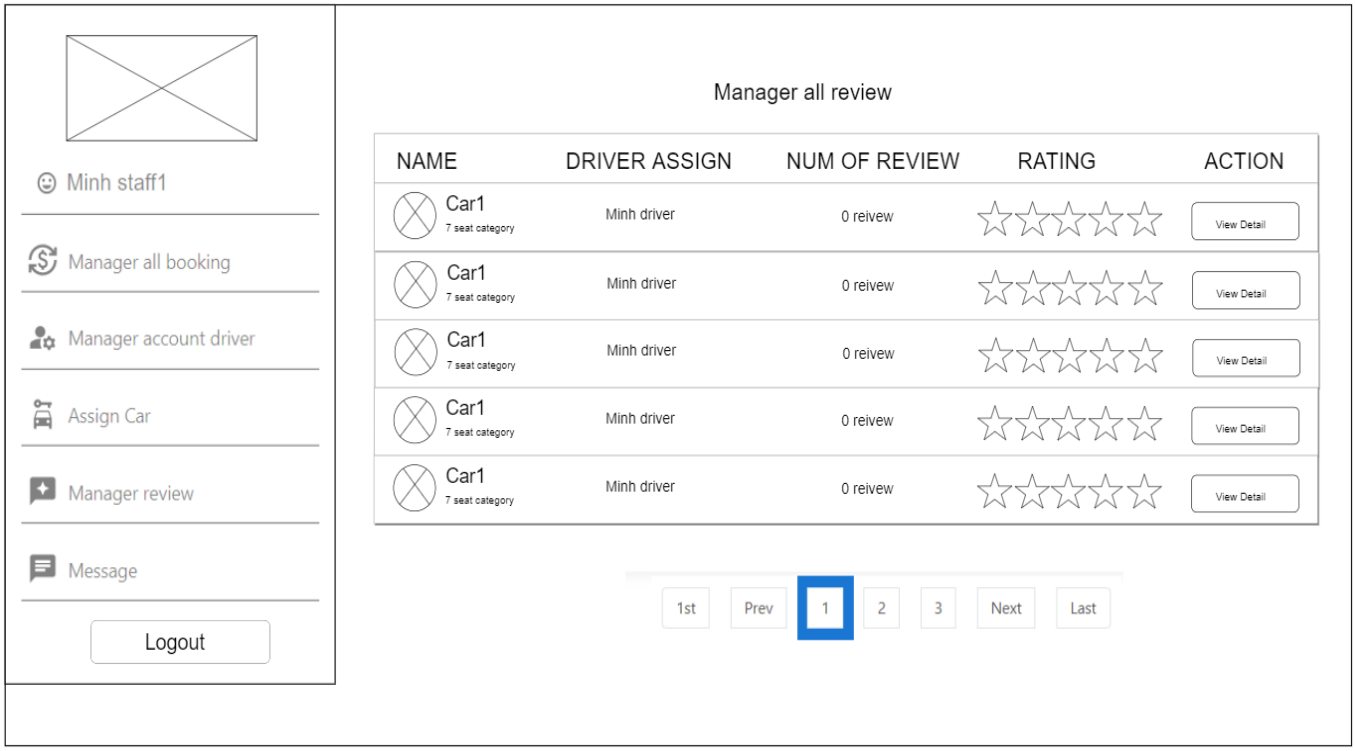



Figure 45: Manager all review for staff wireframe.

Manager review details for staff wireframe.

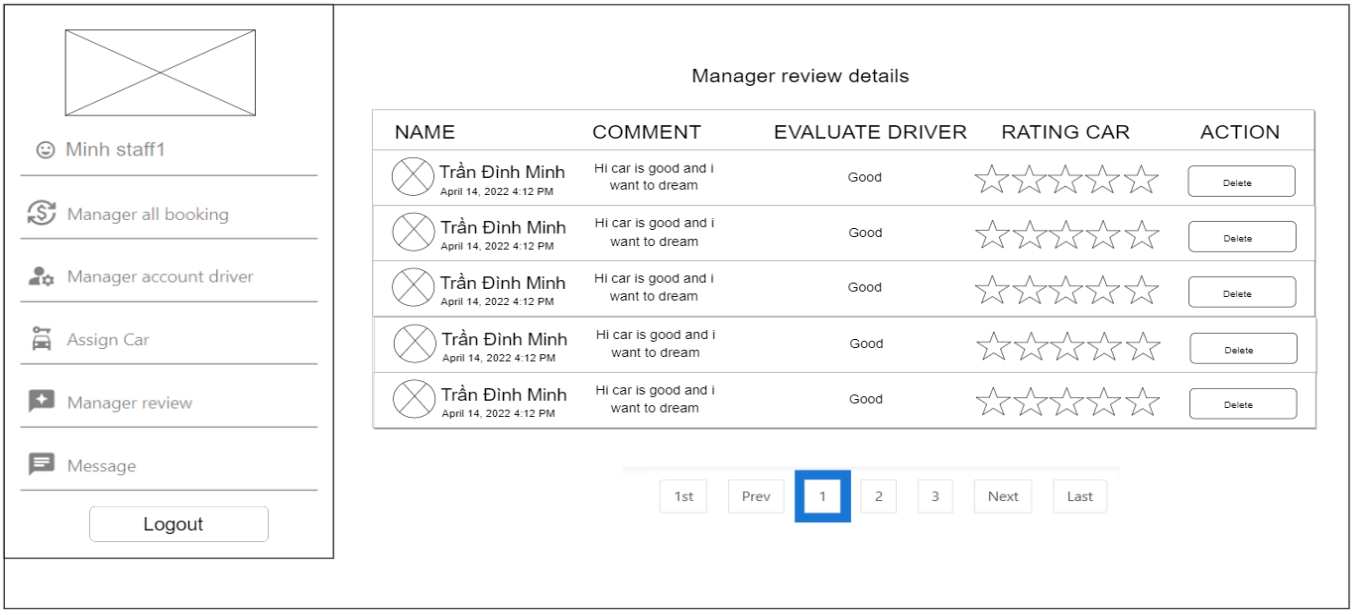



Figure 46: Manager review details for staff wireframe.

Chat message for staff wireframe.

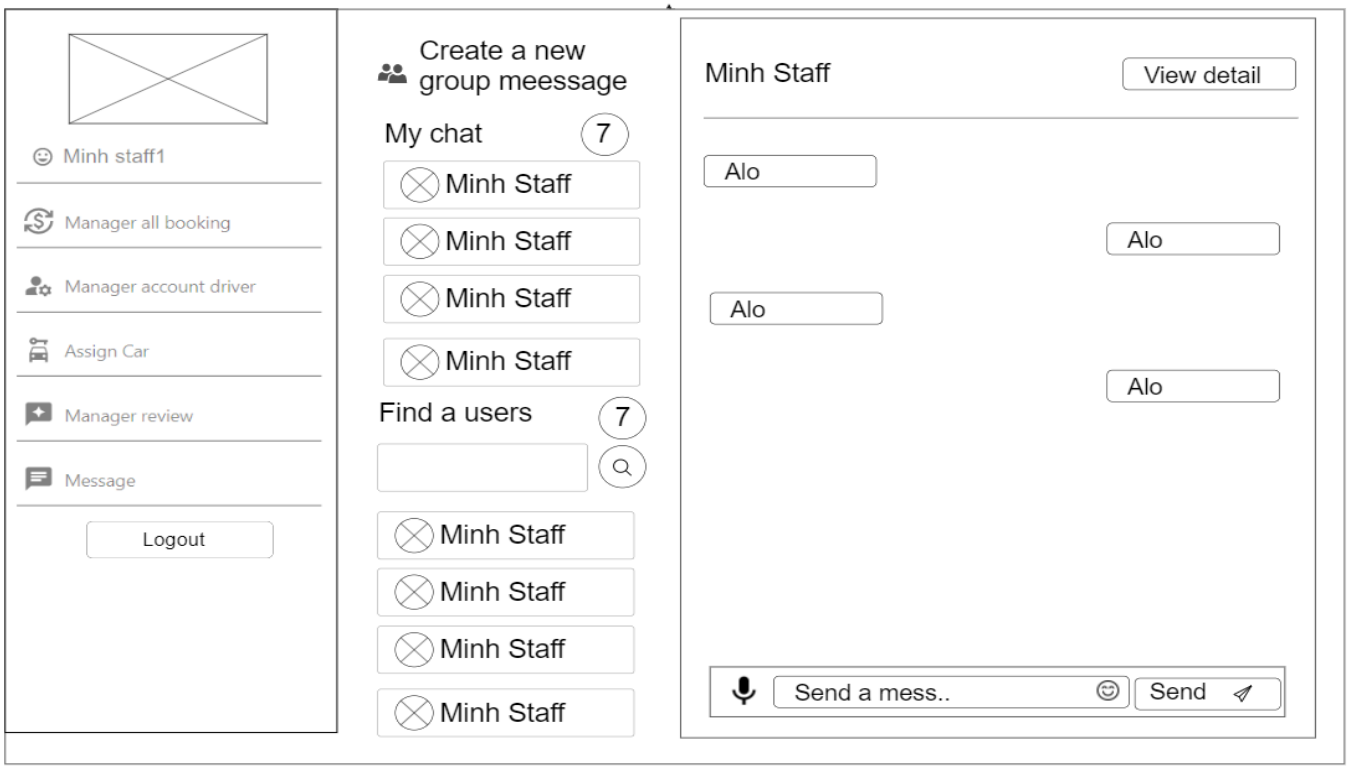



Figure 47: Chat message for staff wireframe.

See detail user chat for staff wireframe.

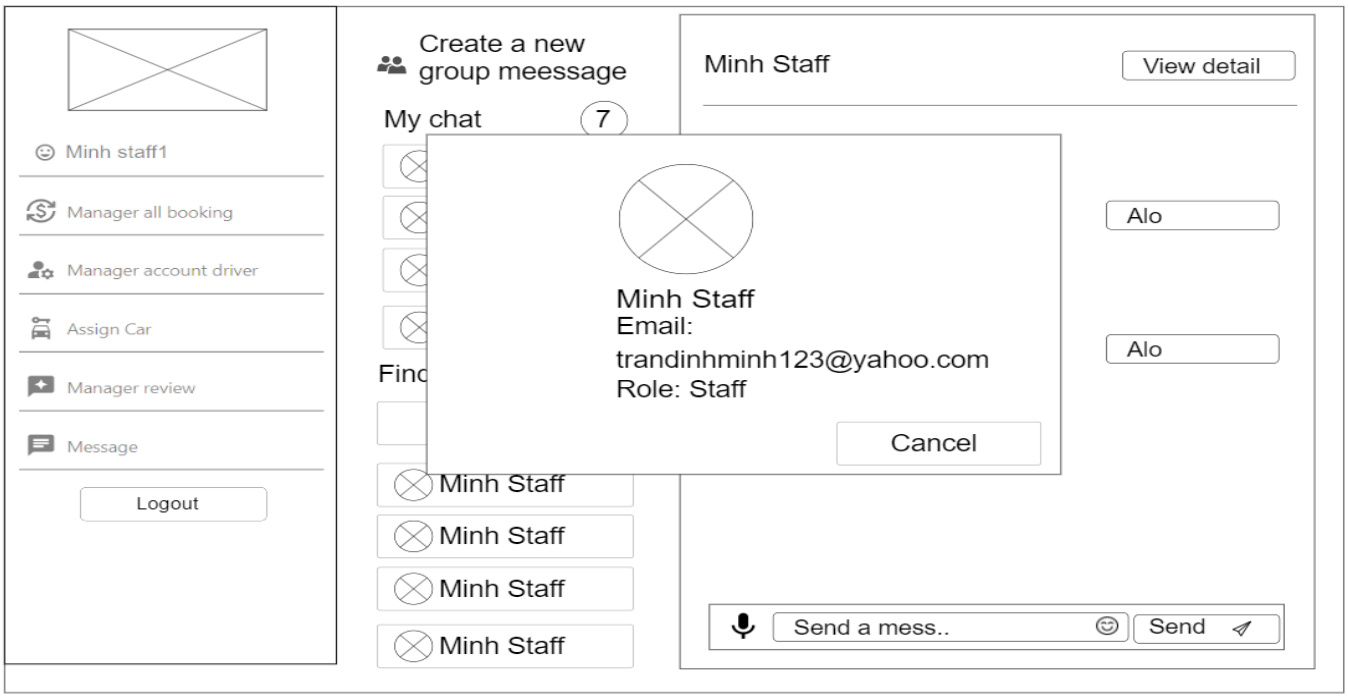



Figure 48: See detail user chat for staff wireframe.

Crate a new group for staff wireframe.

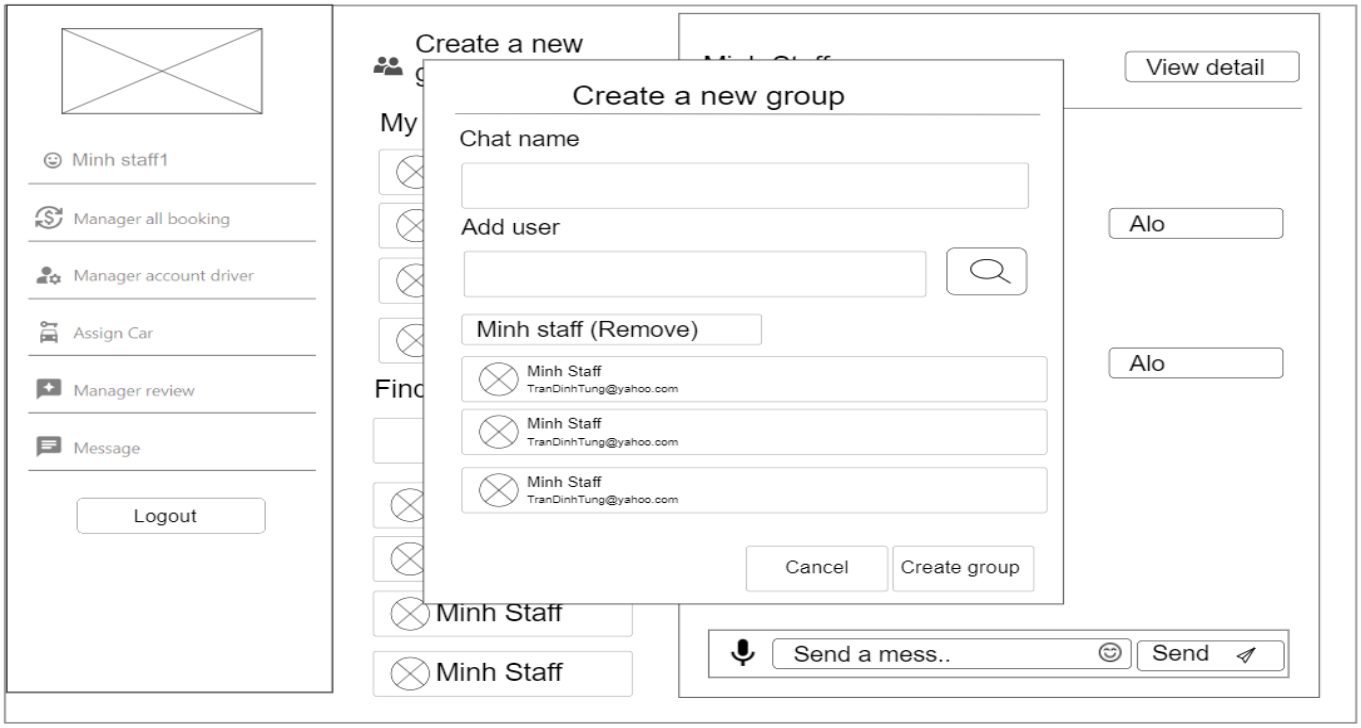



Figure 49: Crate a new group for staff wireframe.

Update name, remove user, leave group, search user for staff wireframe.

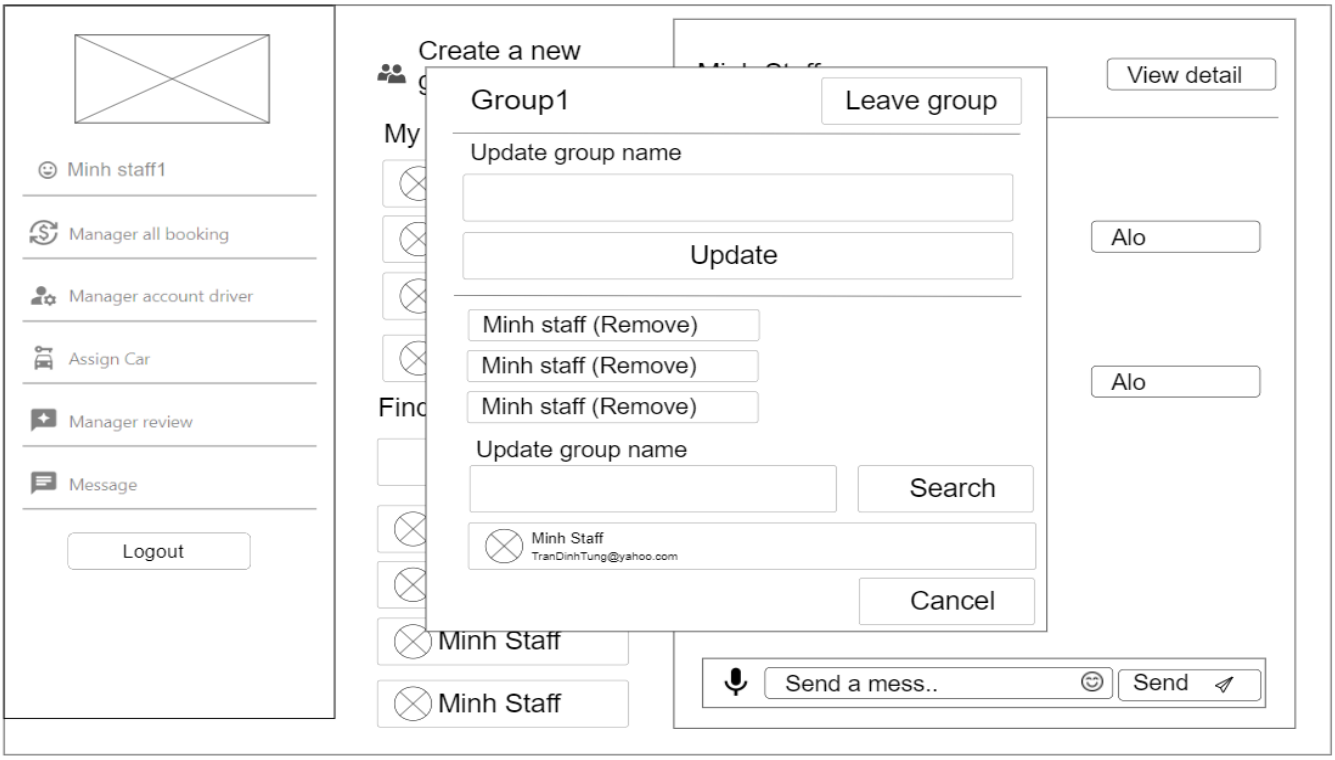



Figure 50: Update name, remove user, leave group, search user for staff wireframe.

### 8.6.5. Wireframes for driver:

Driver sees car assign for driver wireframe.

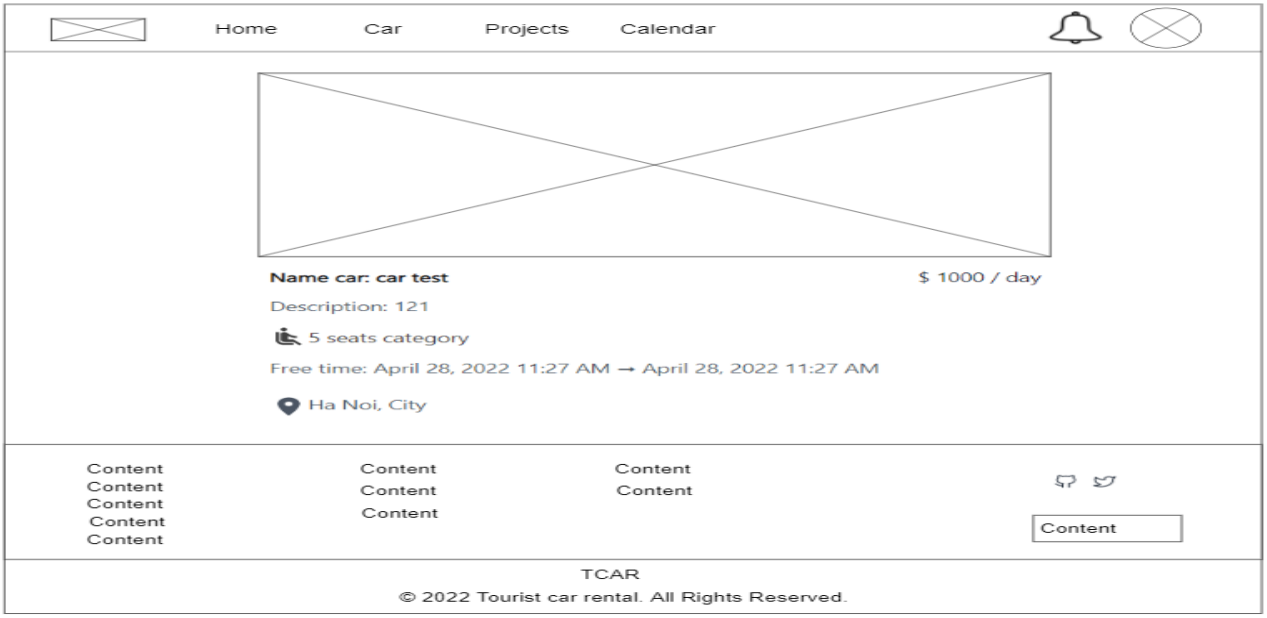



Figure 51: Driver sees car assign for driver wireframe.

Driver sees all user booking for driver wireframe.

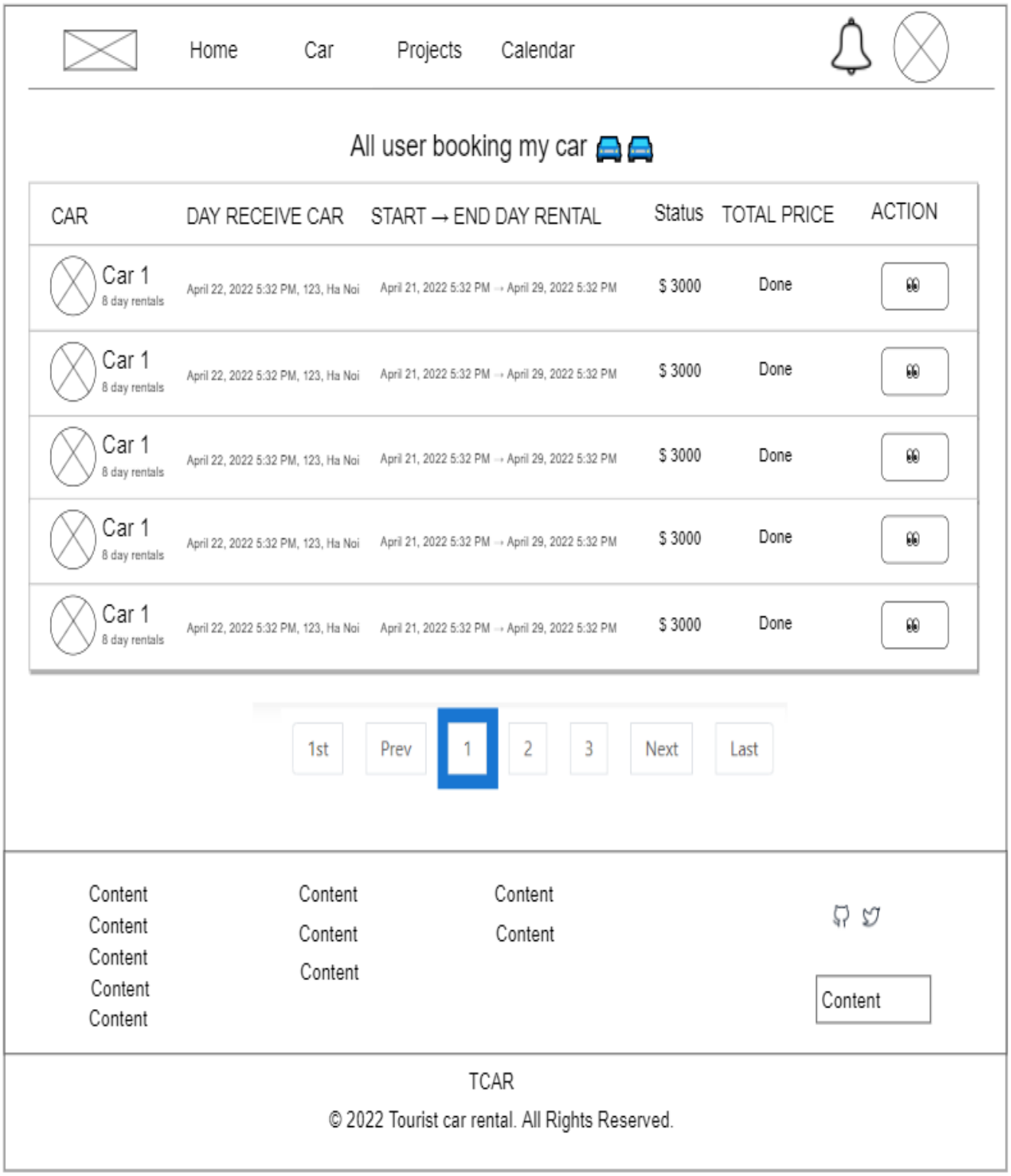



Figure 52: Driver sees all user booking for driver wireframe.

Chat message for driver wireframe.

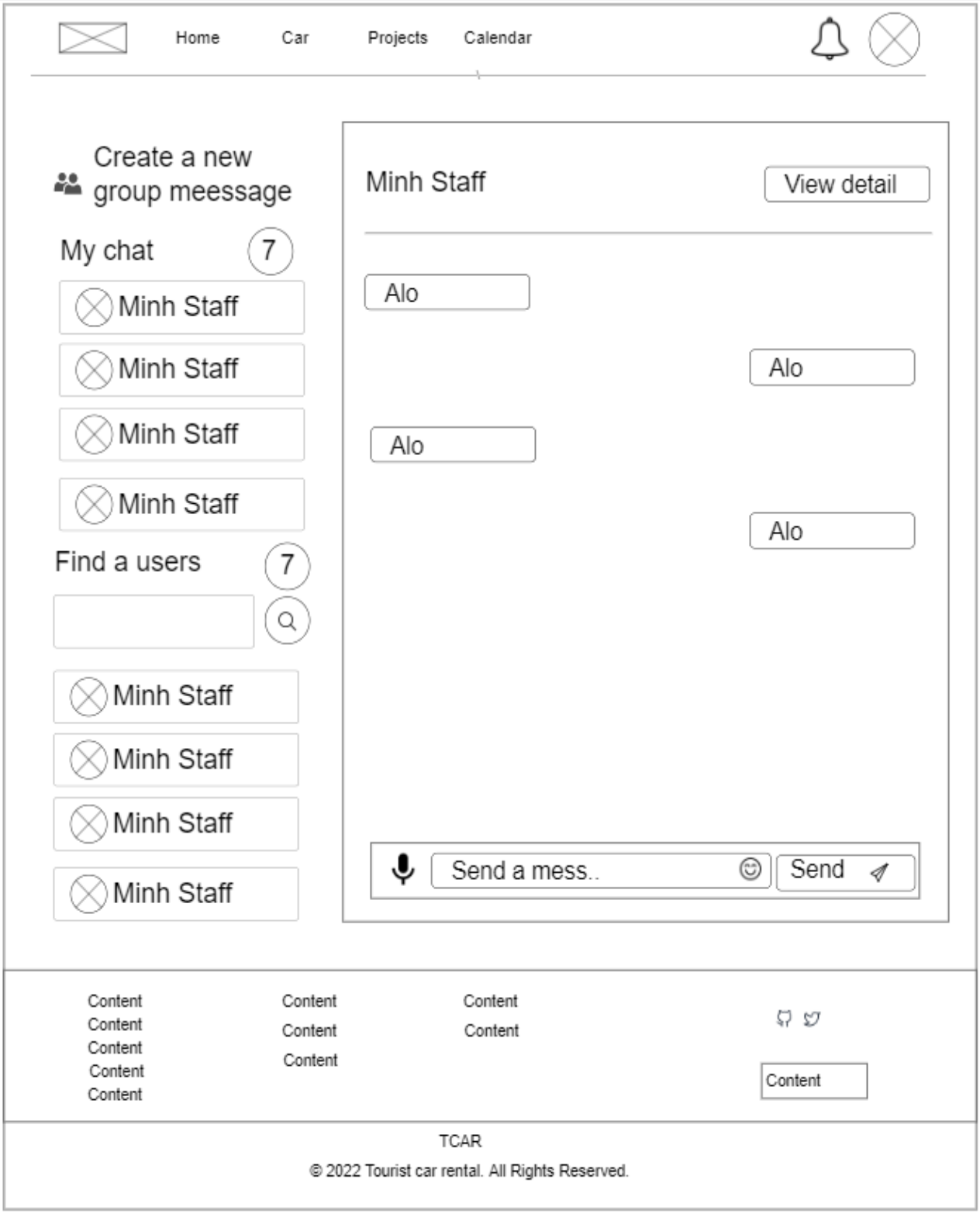



Figure 53: Chat message for driver wireframe.

### 8.6.6. Wireframes for user:

Favorite cart car for user wireframe.

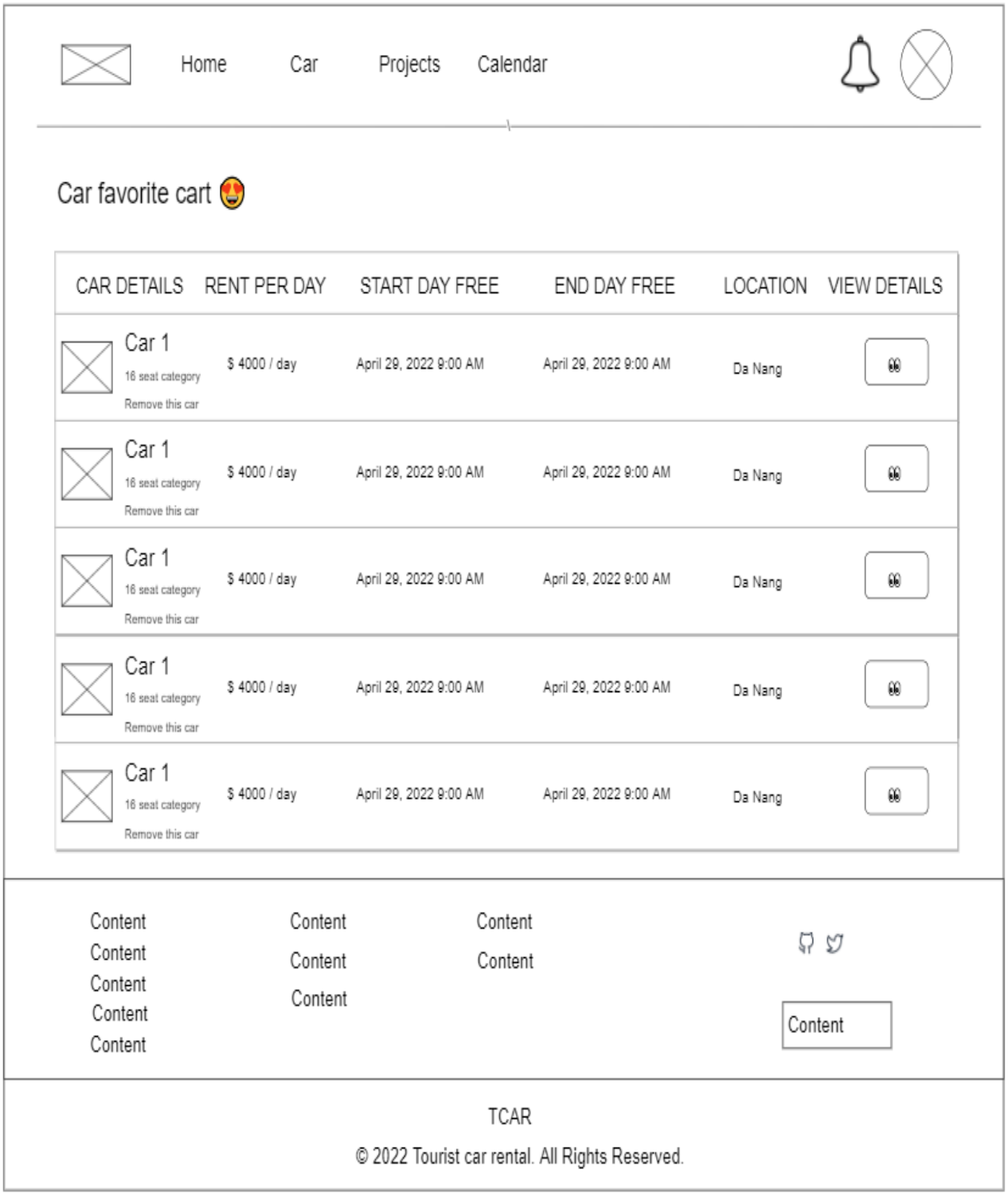



Figure 54: Favorite cart car for user wireframe.

View all car for user wireframe.

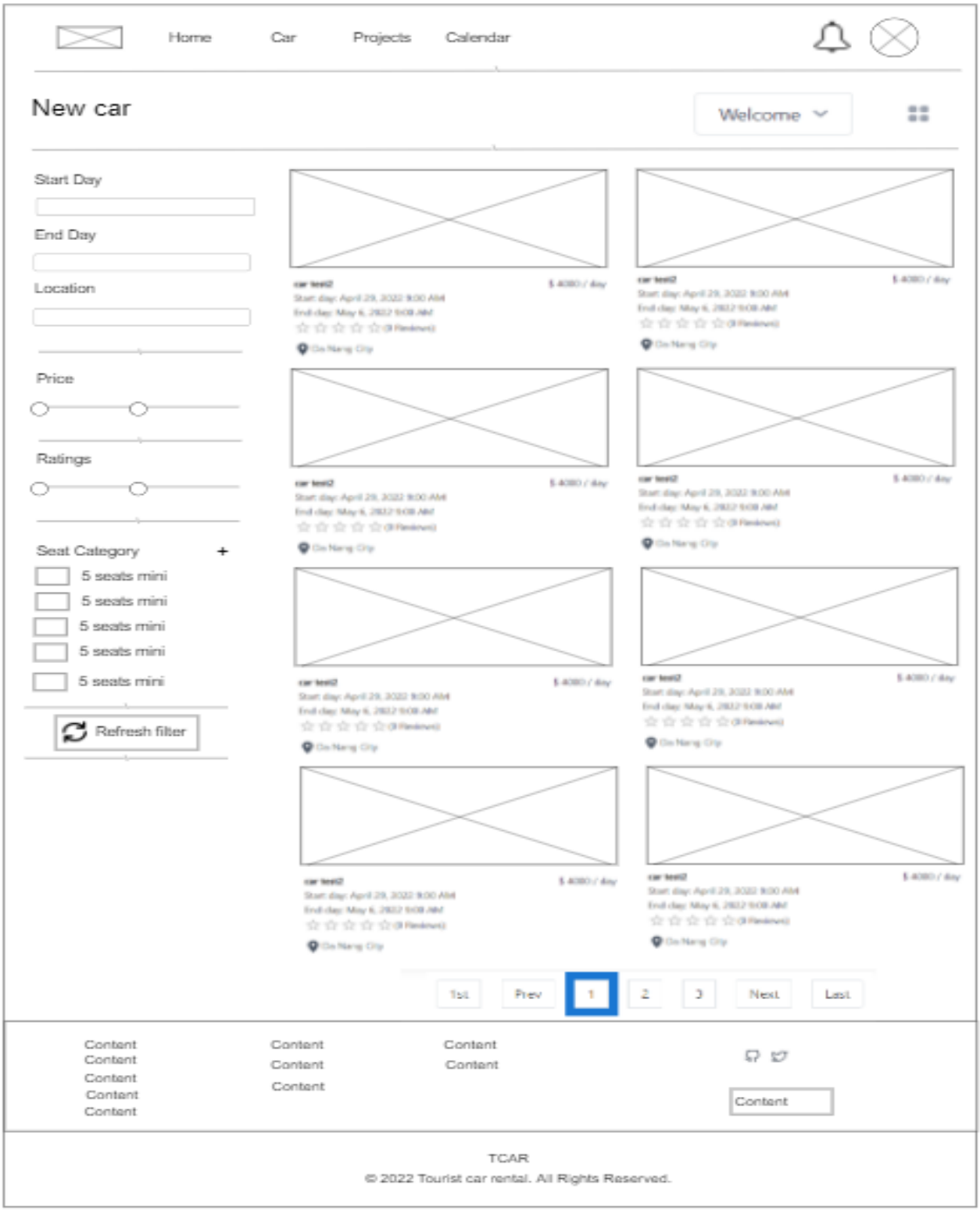



Figure 55: View all car for user wireframe.

View car detail for user wireframe.

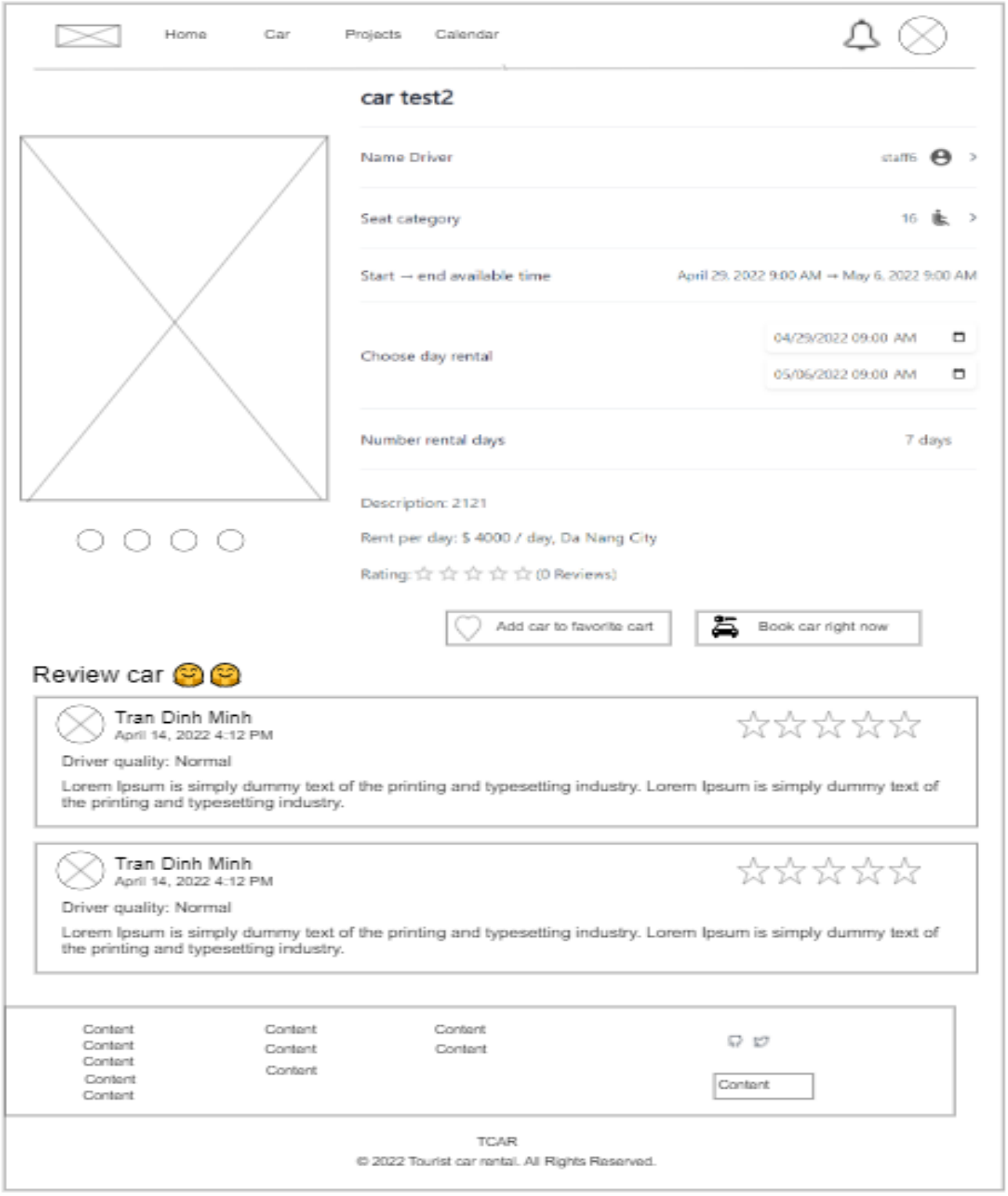



Figure 56: View car detail for user wireframe.

Enter information receive car for user wireframe.


Figure 57: Enter information receive car for user wireframe.

Confirm information booking for user wireframe.

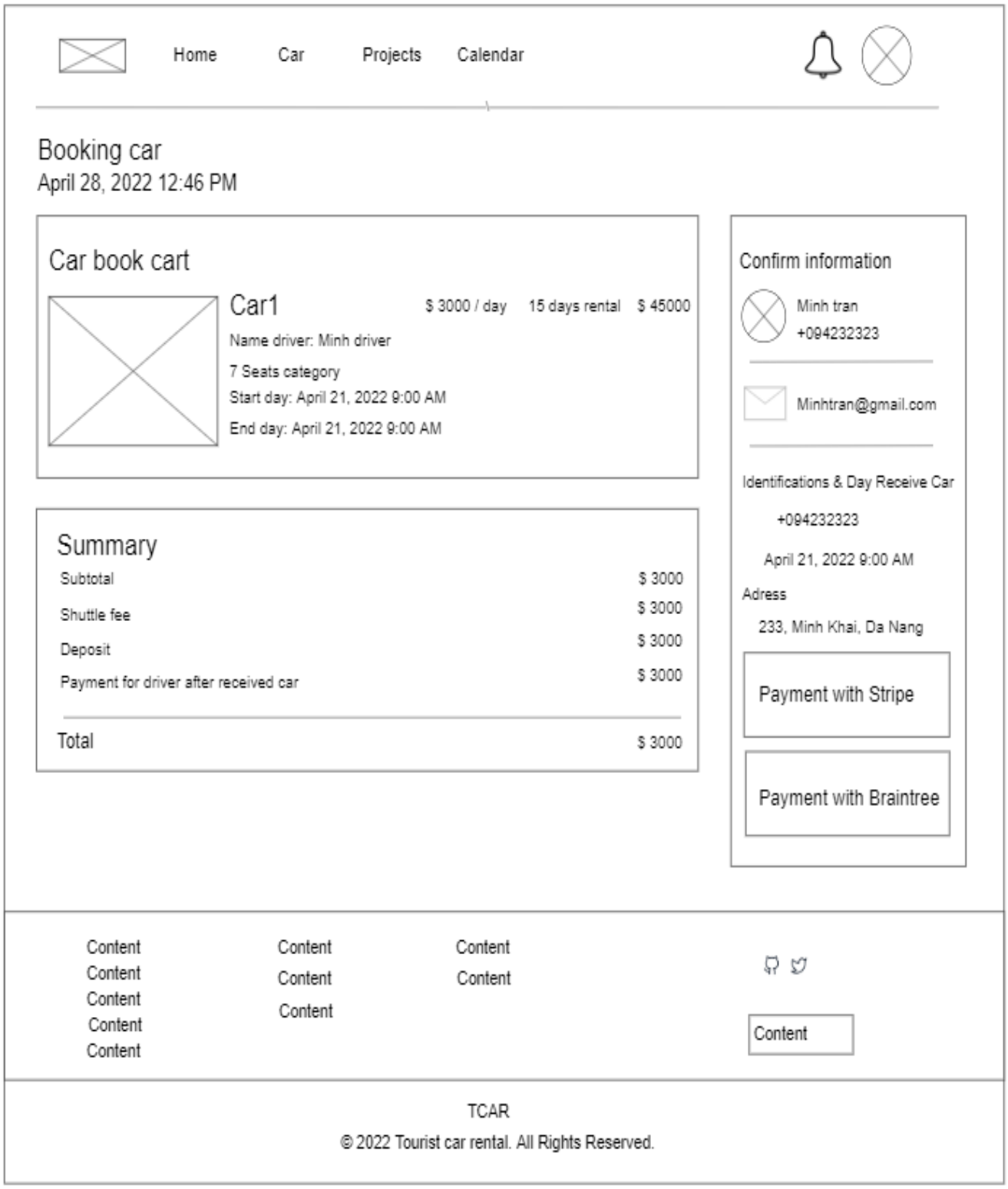



Figure 58: Confirm information booking for user wireframe.

Payment with Braintree for user wireframe.

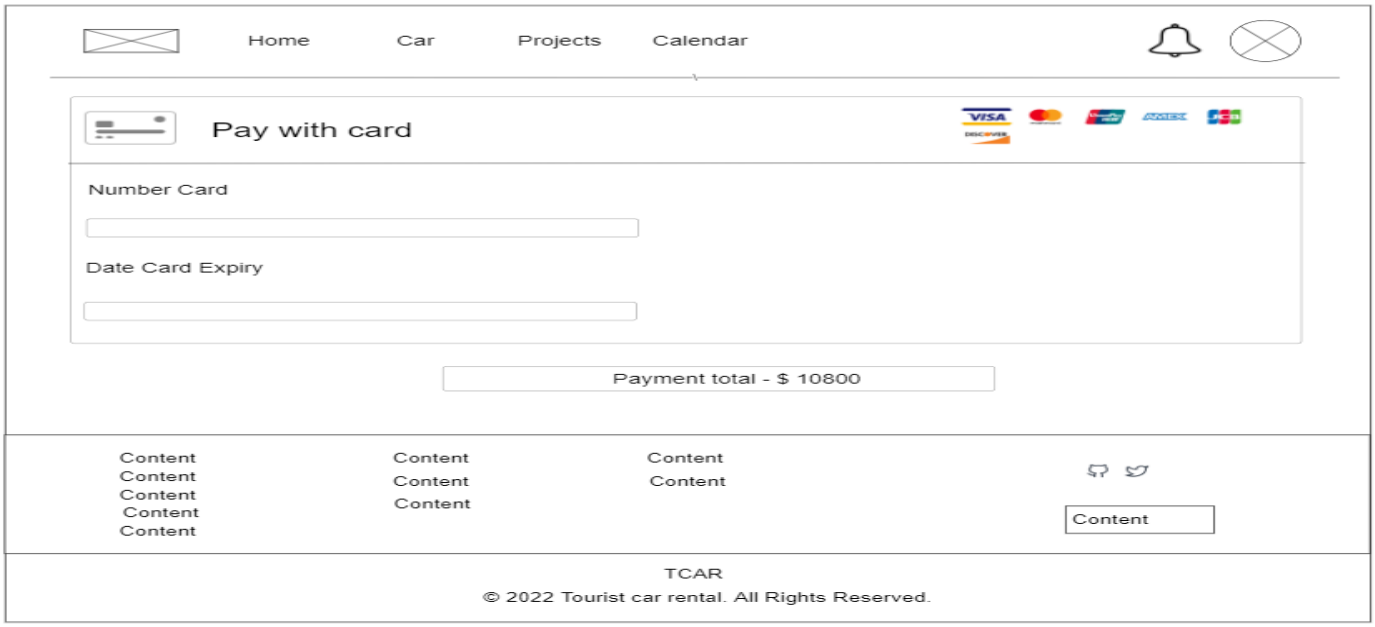



Figure 59: Payment with Braintree for user wireframe.

Payment with Stripe for user wireframe.

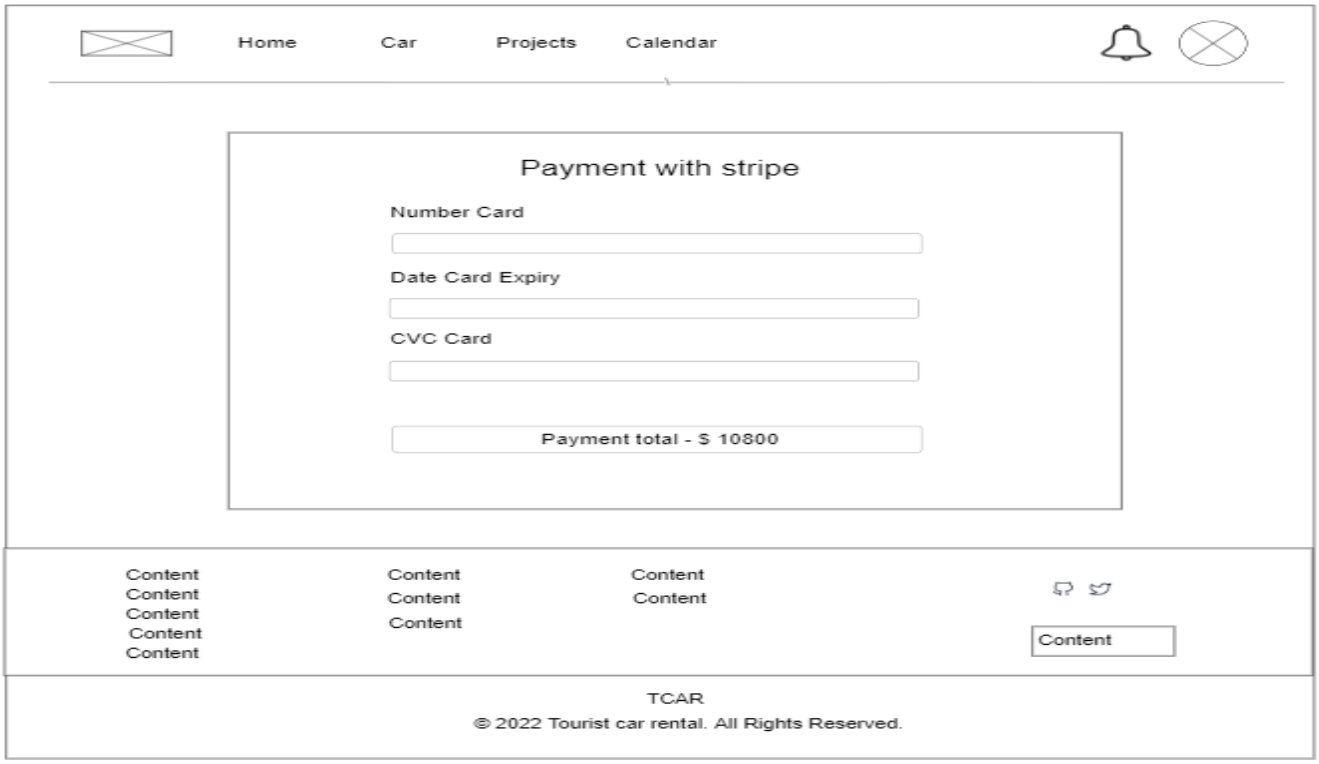



Figure 60: Payment with Stripe for user wireframe.

Payment and booking car success for user wireframe.

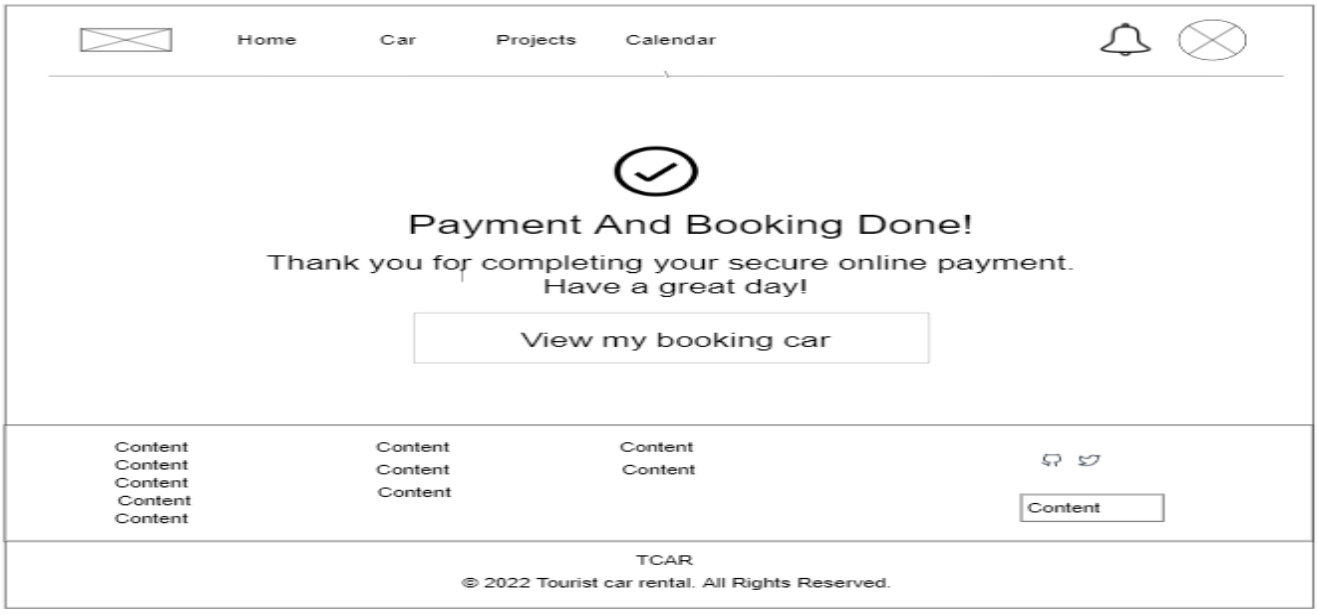



Figure 61: Payment and booking car success for user wireframe.

View my booking (user) for user wireframe.

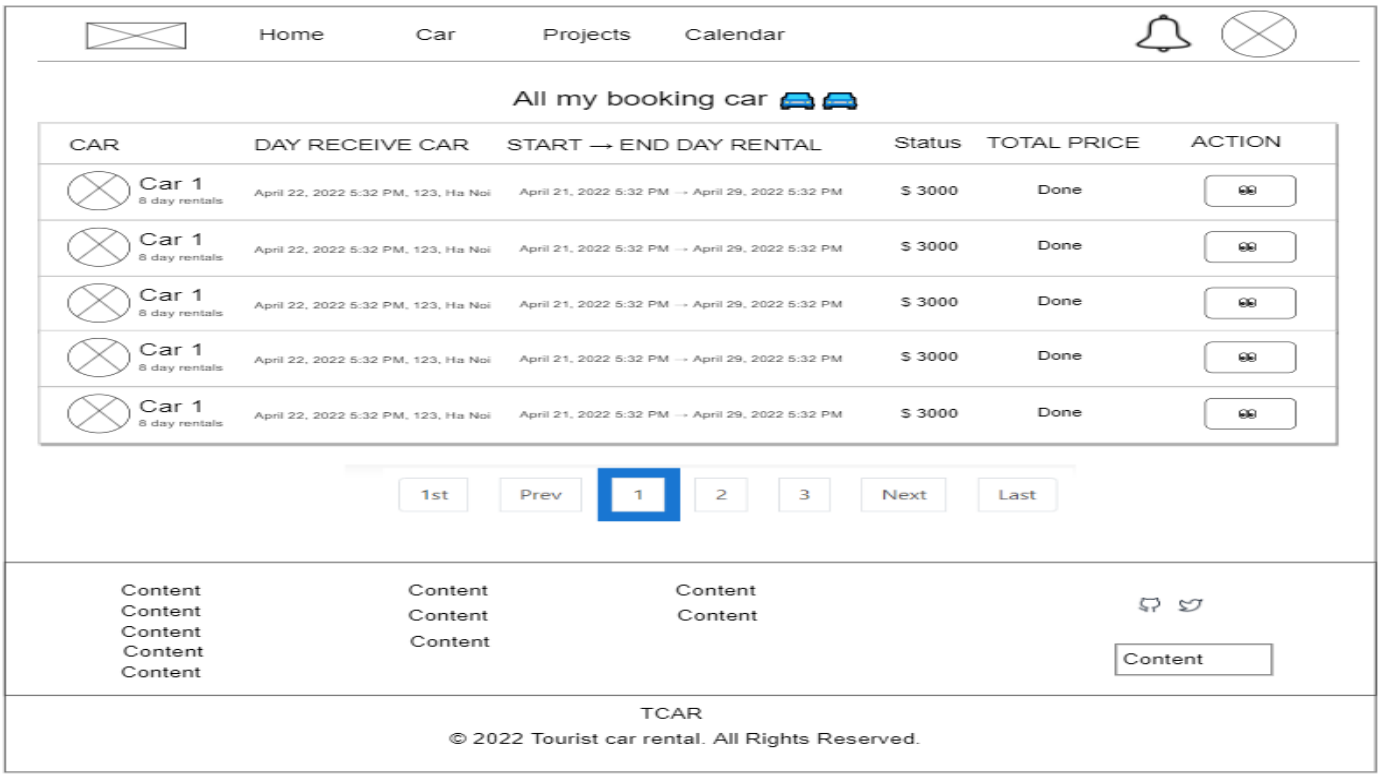



Figure 62: View my booking (user) for user wireframe.

View booking detail have review for user wireframe.

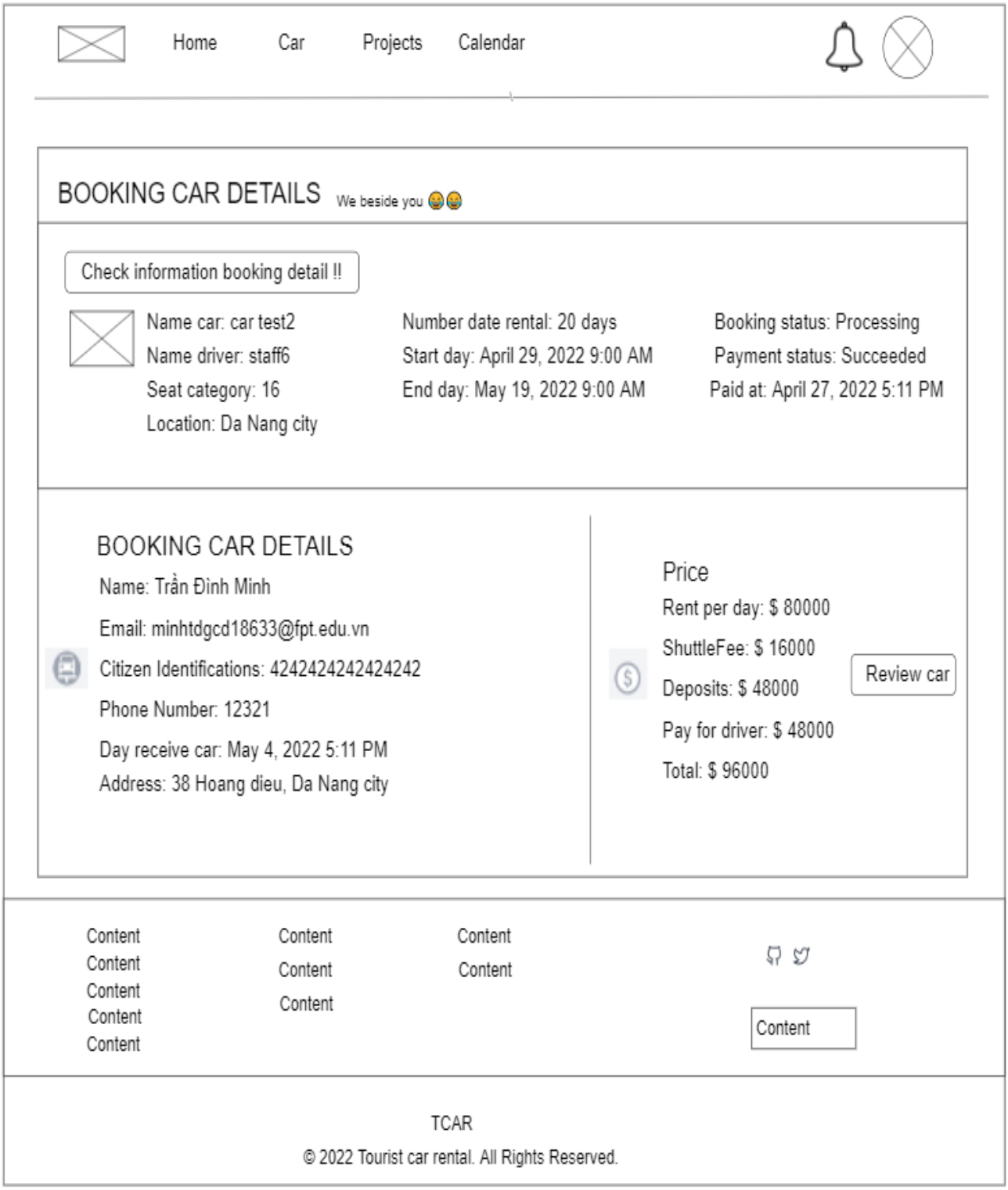



Figure 63: View booking detail have review for user wireframe.

Review car for user wireframe.

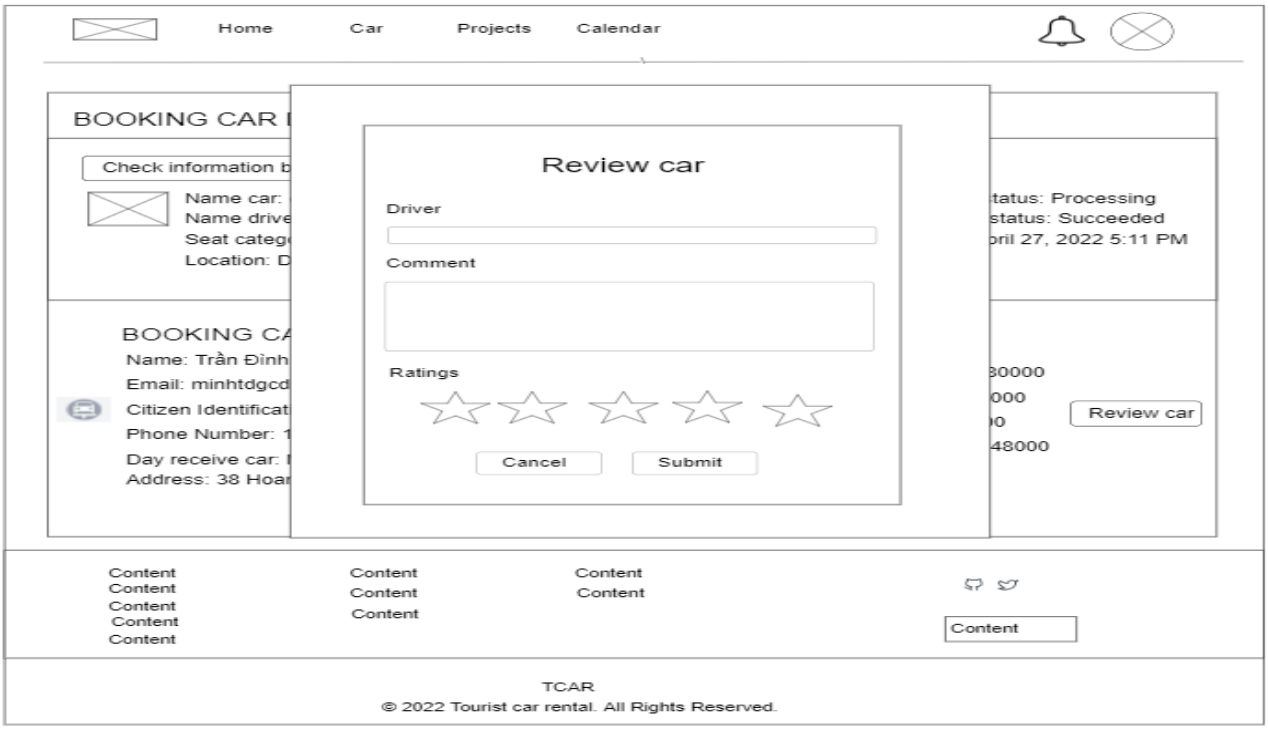



Figure 64: Review car for user wireframe.

### 8.6.7. Wireframes for driver and user:

Profile for driver and user wireframe.

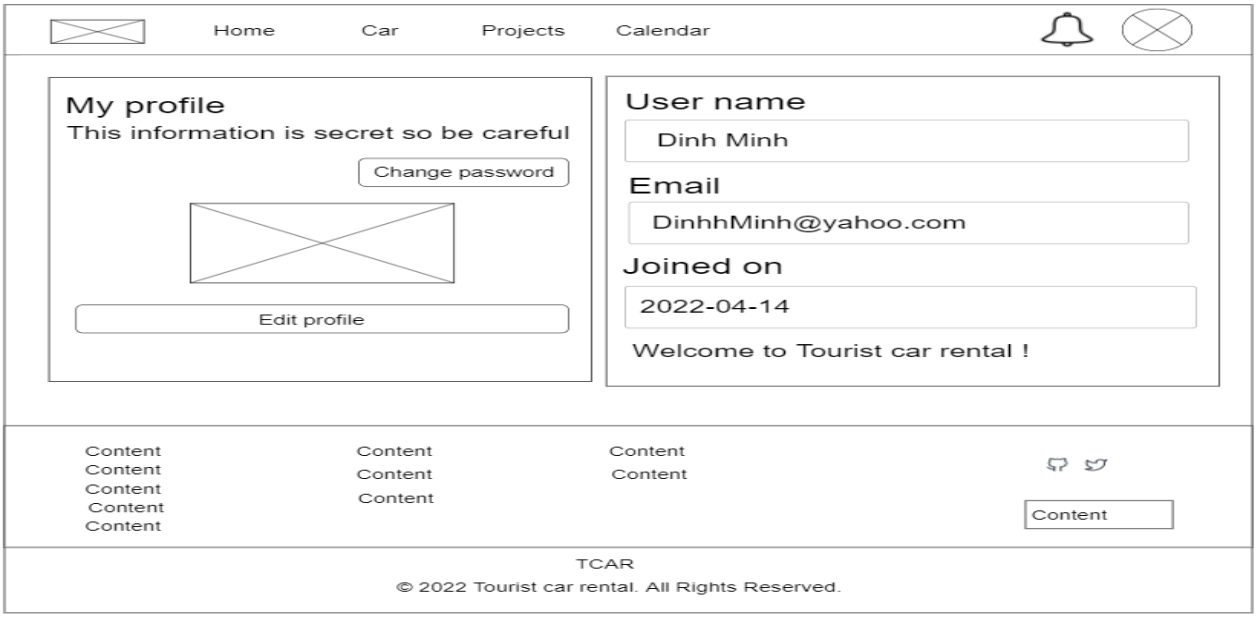



Figure 65: Profile for driver and user wireframe.

Edit profile for driver wireframe.

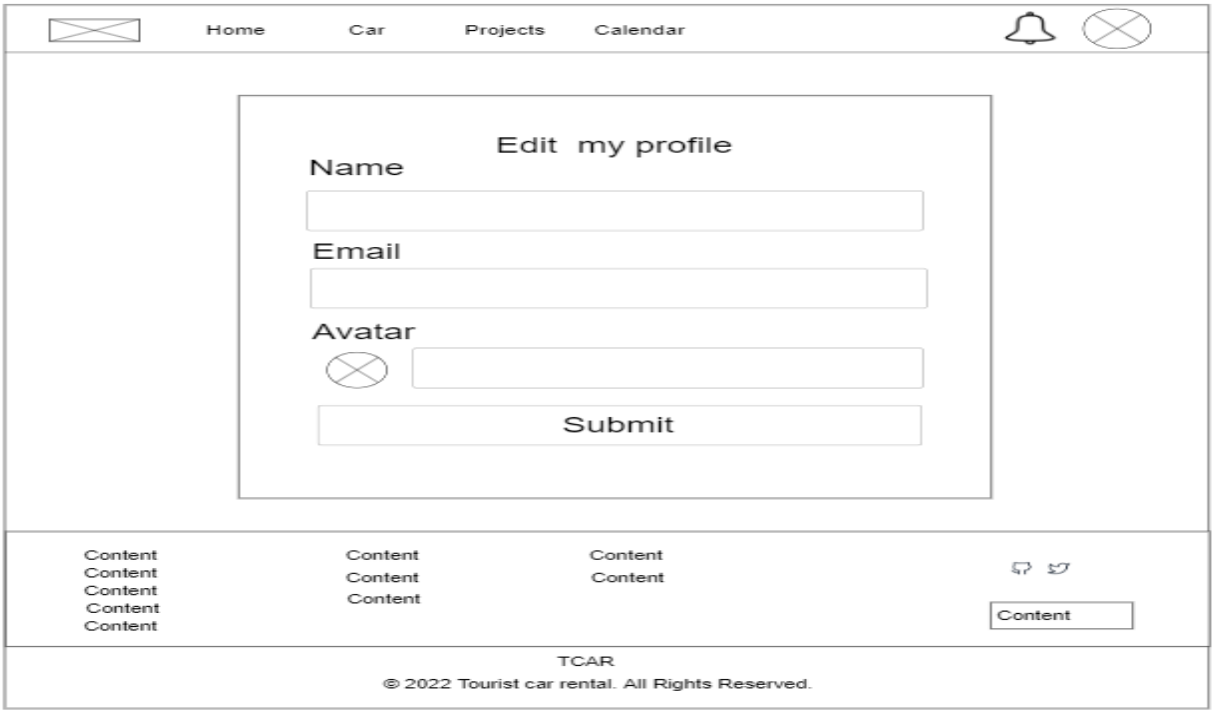



Figure 66: Edit profile for driver wireframe.

Change password for driver wireframe.

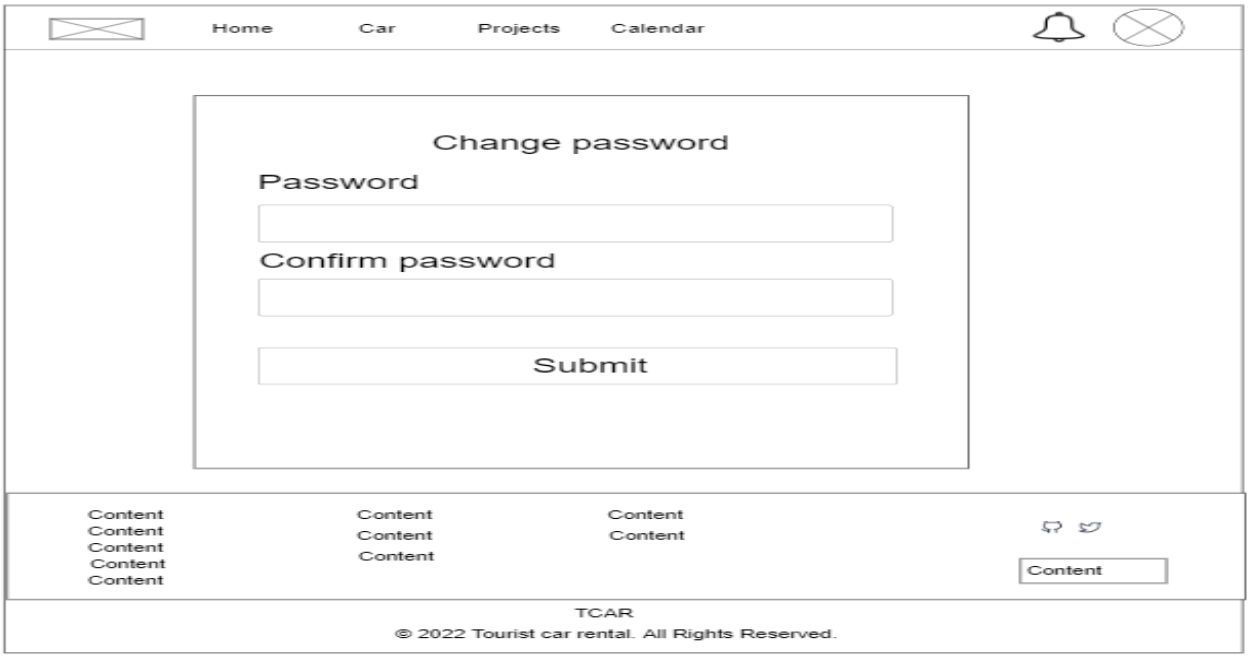



Figure 67: Change password for driver wireframe.

View booking detail not review for user wireframe.

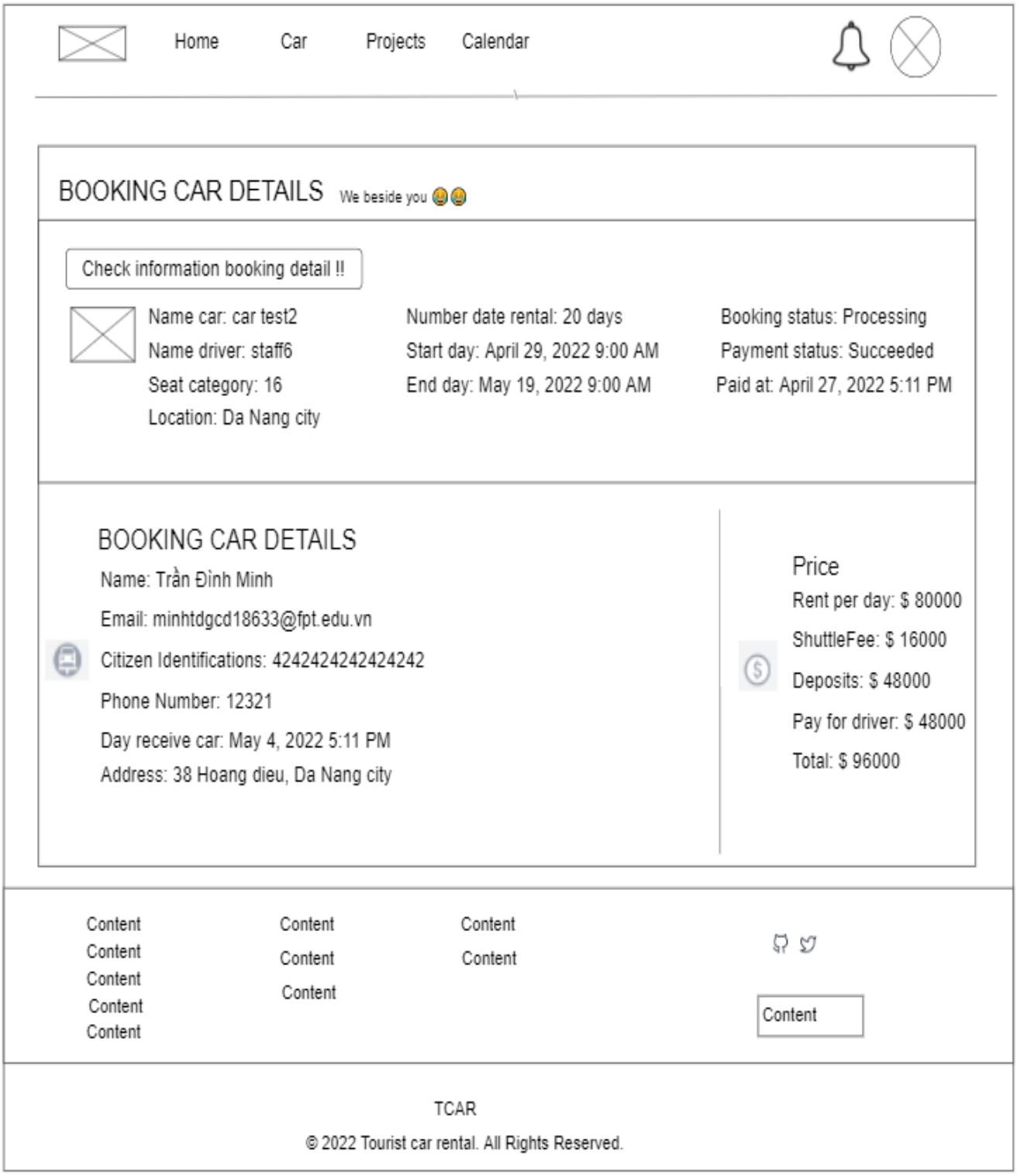



Figure 68: View booking detail not review for user wireframe.

## 8.7. Activity diagram:

Activity diagram for admin:

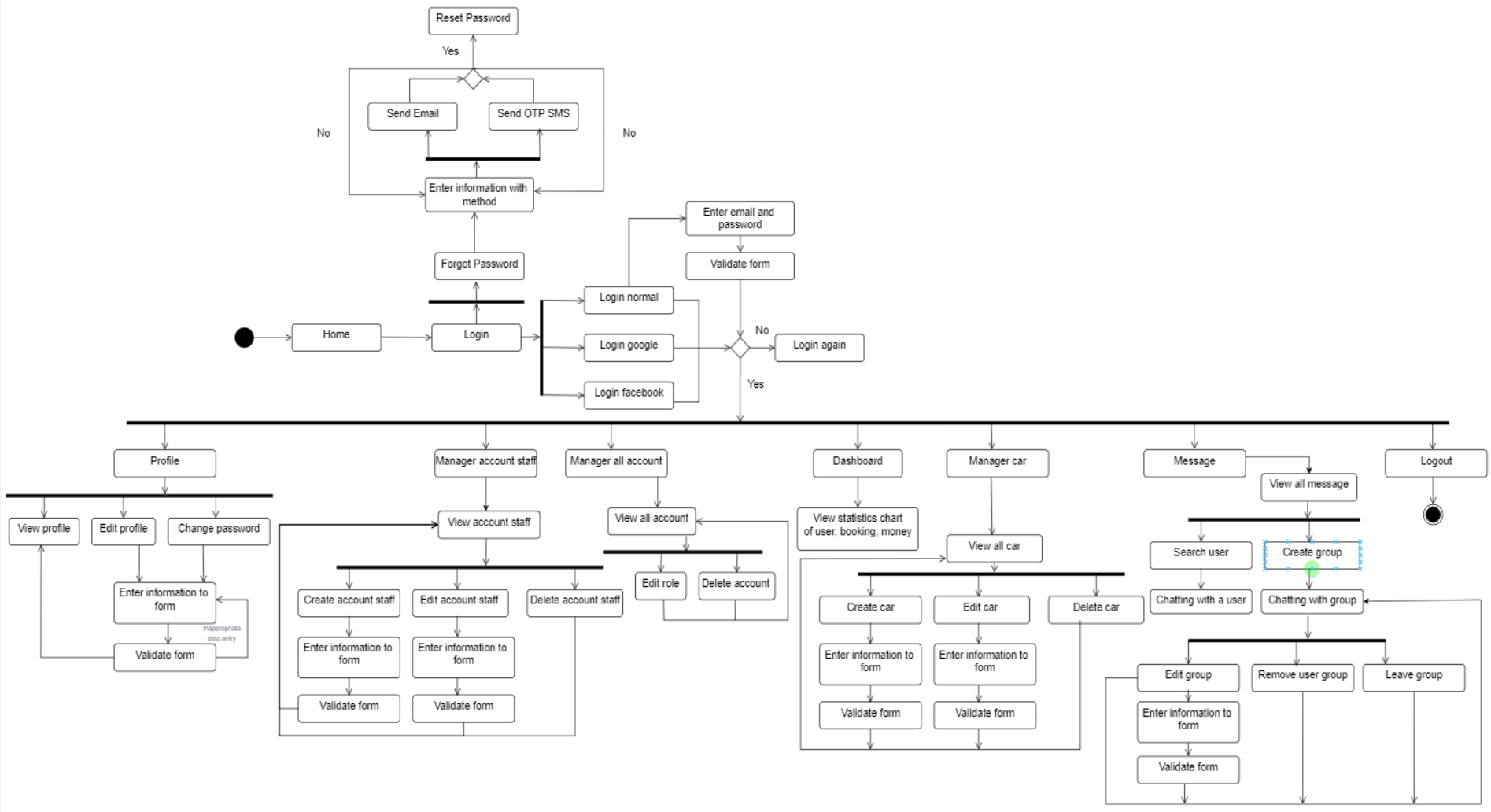



Figure 69: Activity diagram for admin.

Activity diagram for staff:

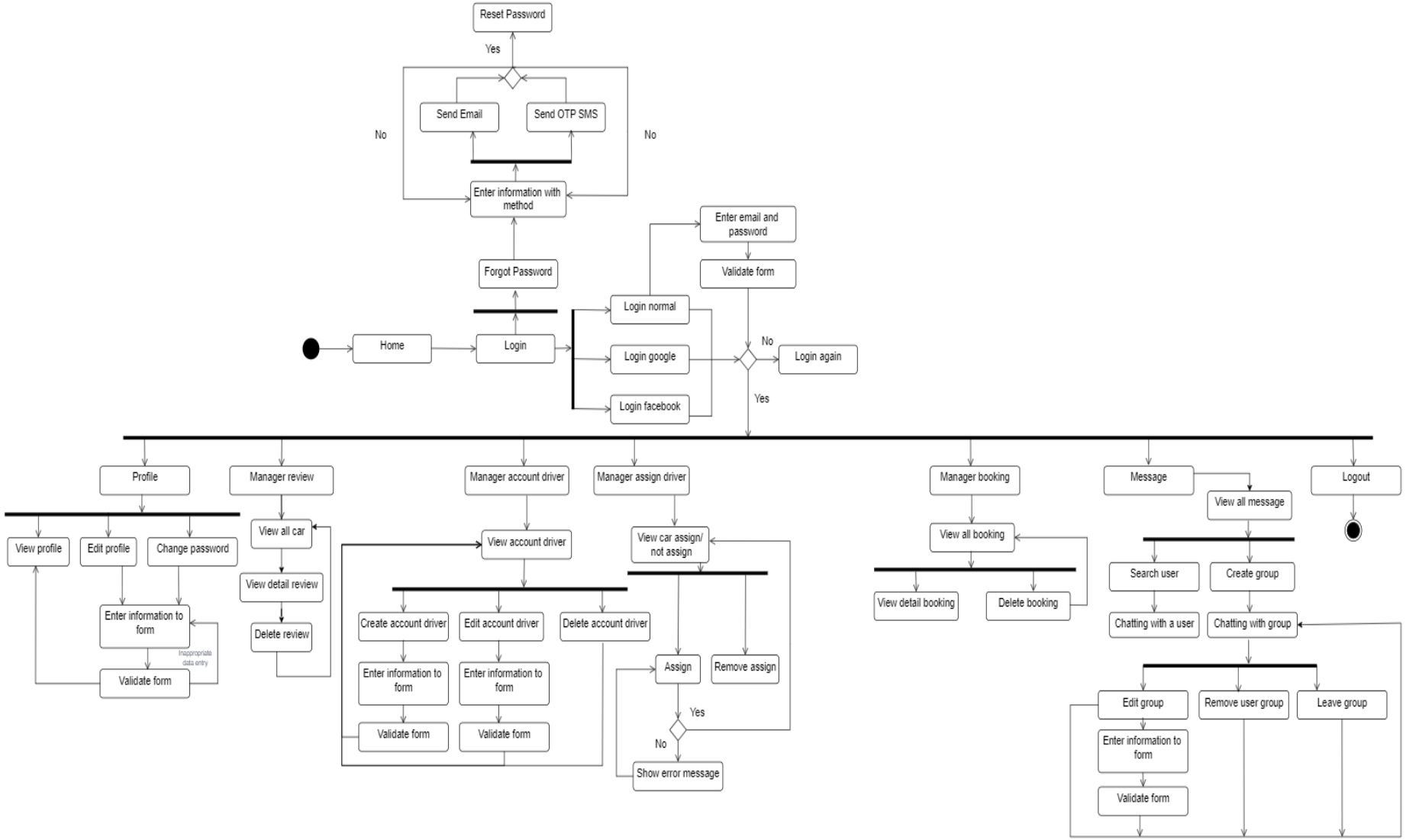



Figure 70: Activity diagram for staff.

Activity diagram for driver:

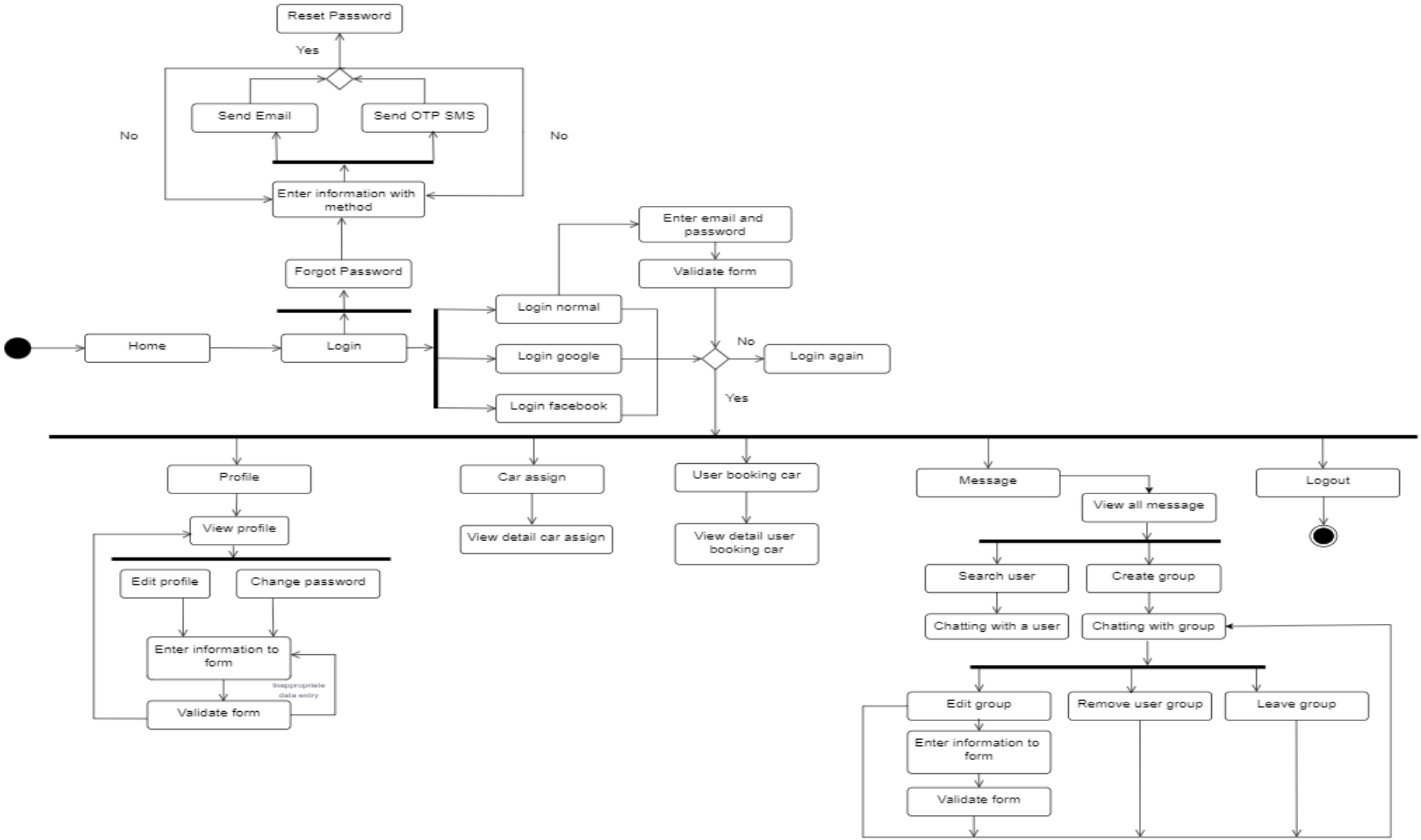



Figure 71: Activity diagram for driver.

Activity diagram for user:

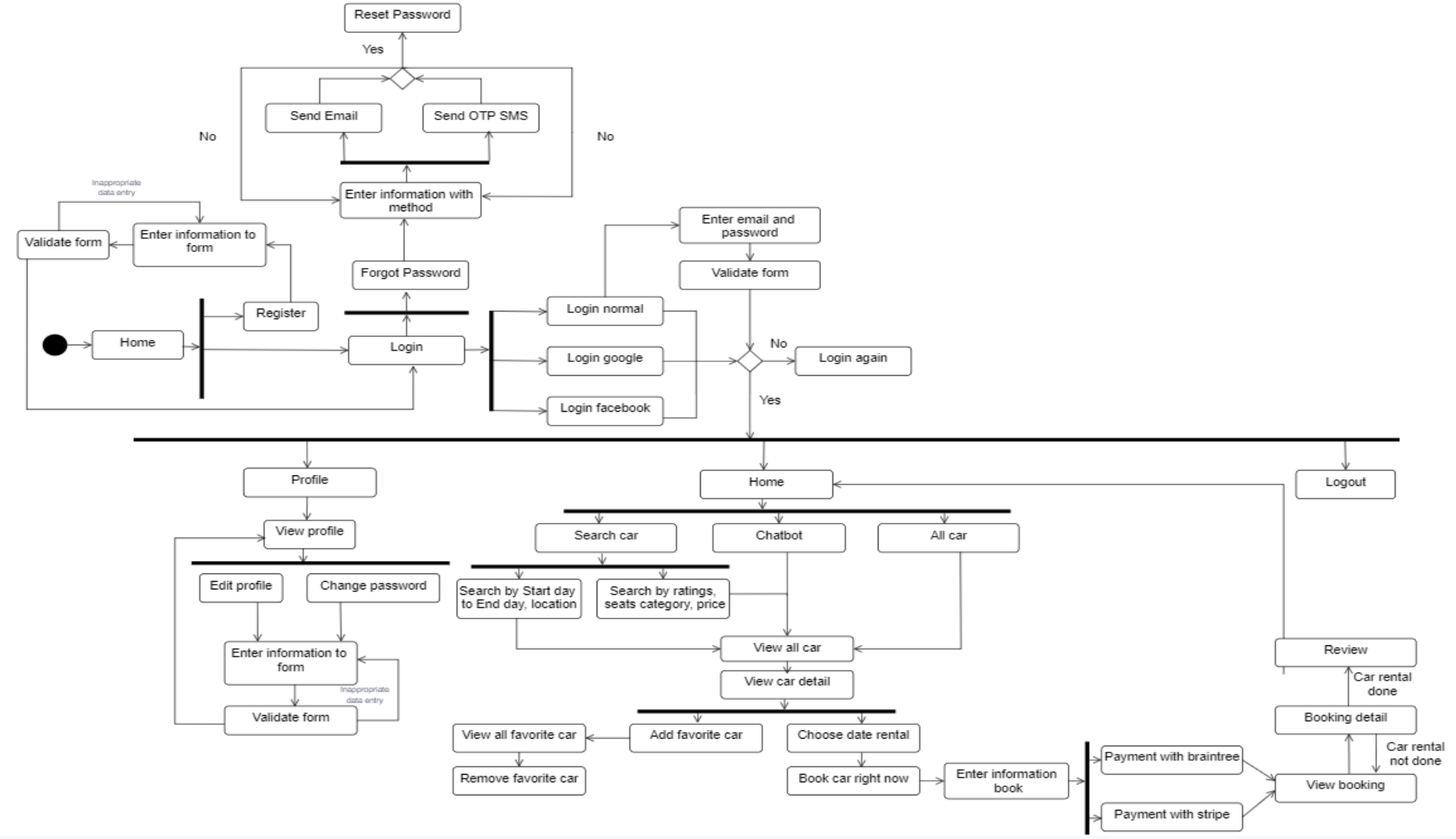



Figure 72: Activity diagram for user.

# 9. Implementation:

## 9.1. Database:

After I have designed the ERD according to section 8.5.1, I will proceed to develop the implementation with each schema will be a collection that I define. I will create 5 schemas respectively booking, car, chat, message, and user.

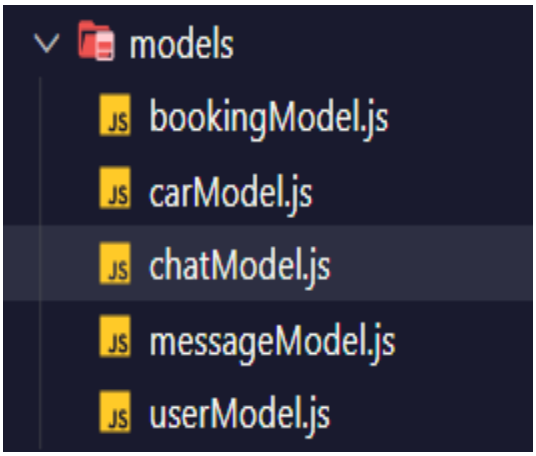



Figure 73: Model database.

Based on the ERD of the user collection, I will create all fields with data types that match ERD, such as name of type string, required input, name must be at least 4 characters and trim is the single existence. The avatar field will have an extra object with 2 public_id and url fields. In the user collection, it is noteworthy that the role field has an enum type for the purpose of identifying 4 roles in the system.

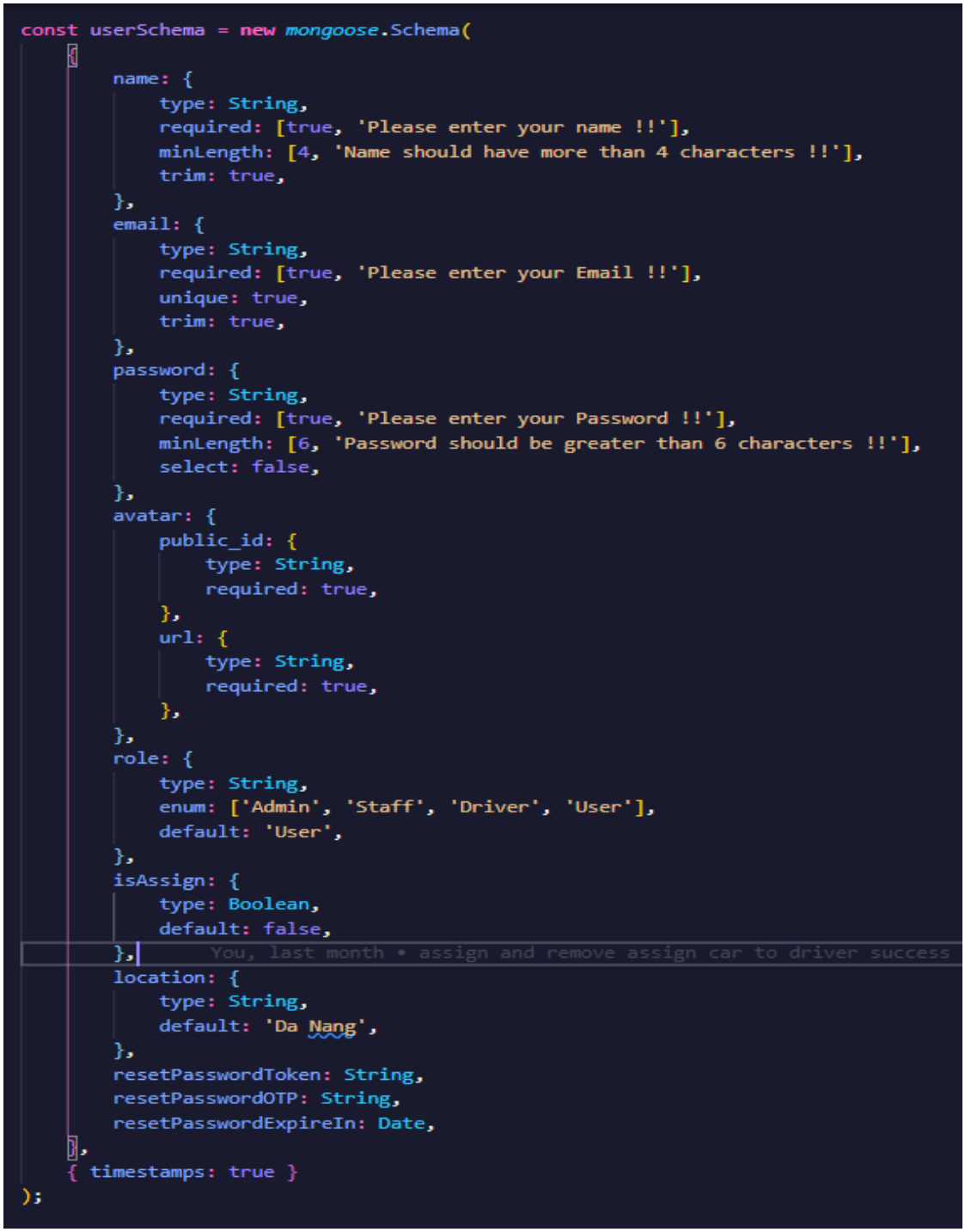



Figure 74: User schema.

Based on the ERD of the collection car, I will create all the fields with the data types that match the ERD. Simple fields are all created like user collections. In this collection, the 3 fields of assigns, rating, and userCreateId will take the ID value from the user collection to perform functions such as who created the table, which driver will be assigned to that car, and who has reviewed that booking.

```
const carSchema = new mongoose.Schema(
    {
        name: {
            type: String,
            required: [true, 'Please Enter car Name !!'],
            trim: true,
        },
        description: {
            type: String,
            required: [true, 'Please Enter car Description !!'],
        },
        images: [
            {
                public_id: {
                    type: String,
                    required: true,
                },
                url: {
                    type: String,
                    required: true,
                },
            },
        ],
        seatsCategory: {
            type: Number,
            enum: [5, 7, 16, 30],
            default: 5,
        },
        ratings: {
            type: Number,
            default: 0,
        },
        location: {
            type: String,
            required: true,
        },
        startDay: {
            type: String,
            required: true,
        },      You, 2 months ago • set up layout create car …
        endDay: {
            type: String,
            required: true,
        },
        rentPerDay: {
            type: Number,
            required: true,
        },
```

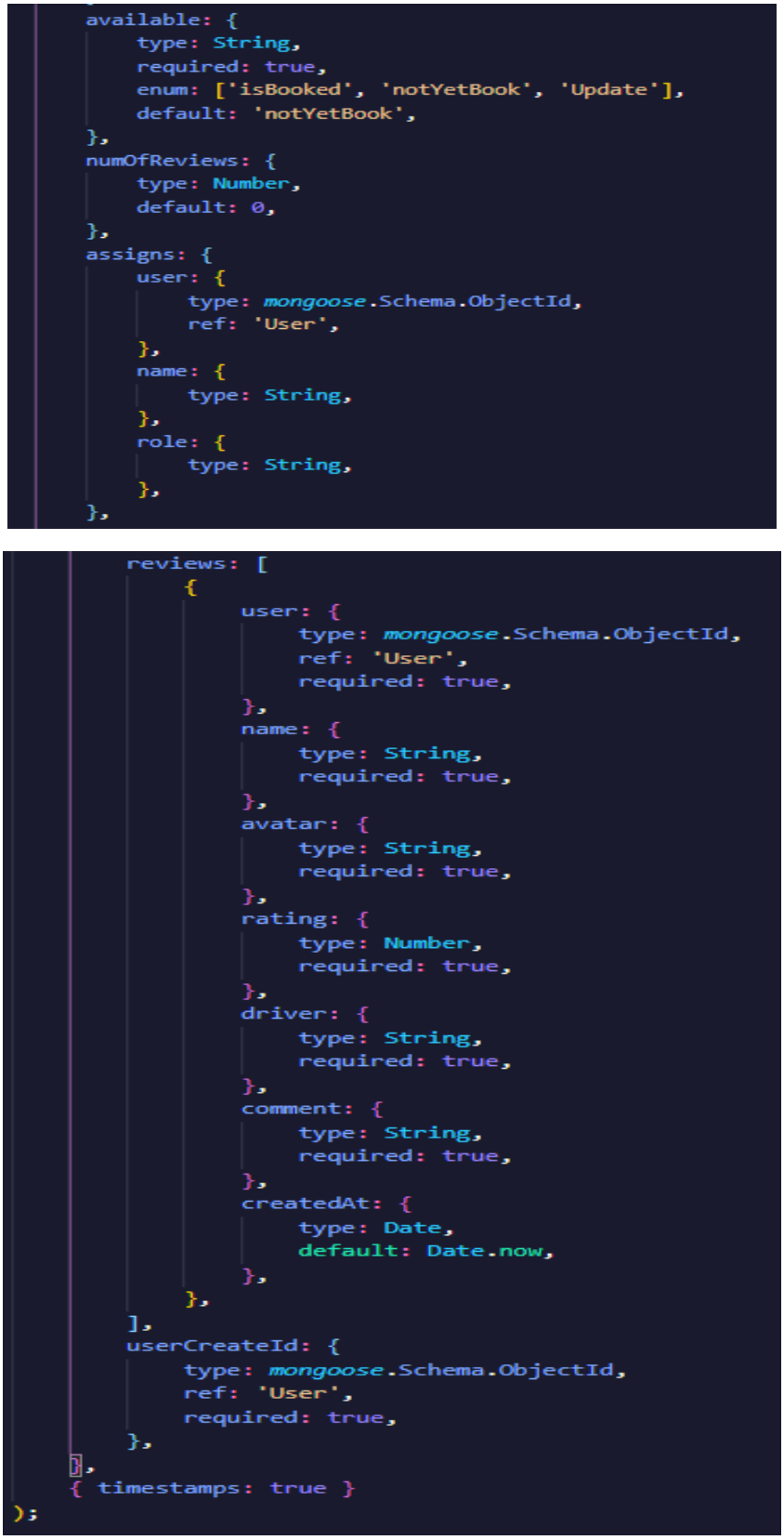



Figure 75: Car schema.

Based on the ERD of the collection booking, I will create all the fields with the data types that match the ERD. In this collection, the generated fields are quite complex, the receivingCarTo, userBooking fields are all accompanied by an object containing the necessary information for the functions that I will develop. The necessary fields that use the fields from the user and car collections are all set up by the schema that takes the ID of each of those collections. This collection is mainly for storing car booking user information, adding cars to favorite cart and storing payment information.

```
const bookingSchema = new mongoose.Schema(
    {
        receivingCarTo: {
            citizenIdentifications: {
                type: Number,
                required: true,
            },
            phoneNum: {
                type: Number,
                required: true,
            },
            day: {
                type: String,
                required: true,
            },
            address: {
                type: String,
                required: true,
            },
            location: {
                type: String,
                required: true,
            },
        },
        bookCars: [
            {
                name: {
                    type: String,
                    required: true,
                },
                nameDriver: {
                    type: String,
                    required: true,
                },
                seatsCategory: {
                    type: Number,
                    required: true,
                },
                rentPerDay: {
                    type: Number,
                    required: true,
                },
                quantity: {
                    type: Number,
                    required: true,
                },
                image: {
                    type: String,
                    required: true,
                },
                startDay: {
                    type: String,
                    required: true,
                },
                endDay: {
                    type: String,
                    required: true,
                },
                location: {
                    type: String,
                    required: true,
                },
```

```
    bookCars: [
        {
            name: {
                type: String,
                required: true,
            },
            nameDriver: {
                type: String,
                required: true,
            },
            seatsCategory: {
                type: Number,
                required: true,
            },
            rentPerDay: {
                type: Number,
                required: true,
            },
            quantity: {
                type: Number,
                required: true,
            },
            image: {
                type: String,
                required: true,
            },
            startDay: {
                type: String,
                required: true,
            },
            endDay: {
                type: String,
                required: true,
            },
            location: {
                type: String,
                required: true,
            },
            driverID: {
                type: mongoose.Schema.ObjectId,
                ref: 'User',
                required: true,
            },
            car: {
                type: mongoose.Schema.ObjectId,
                ref: 'Car',
                required: true,
            },
        },
    ],
    userBooking: {
        user: {
            type: mongoose.Schema.ObjectId,
            ref: 'User',
        },
        nameUser: {
            type: String,
        },
        email: {
            type: String,
        },
    },
```

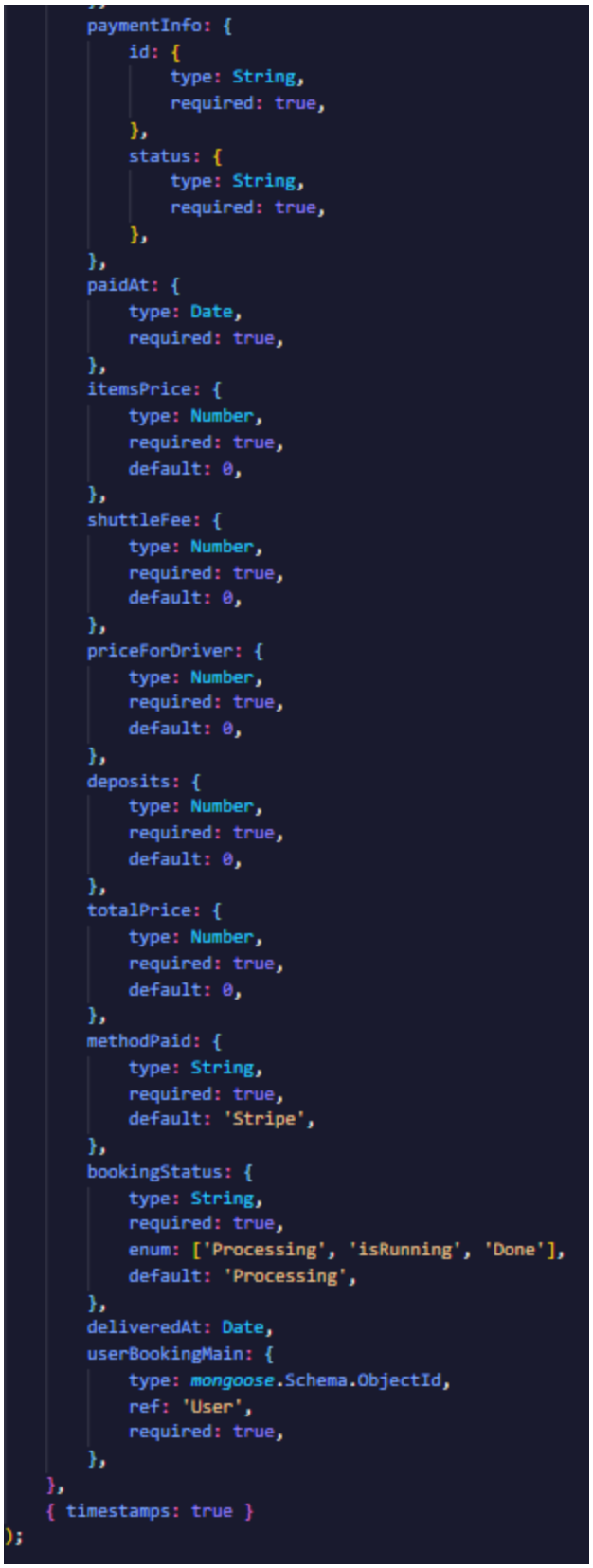



Figure 76: Booking schema.

Based on the chat collection ERD, I will create all the fields with data types that match the ERD. In this collection, the chatName field will be the name of the group chat, the isGroupChat field to determine whether it is a group chat or a user chats with another user. The users and groupCreatedBy fields are assigned with the ID of the User schema for the purpose of identifying the group creator and the person who is logged in on the system to find other users to chat with them. The lastestMessage field will be assigned the ID of the Message schema.

```
const chatSchema = new mongoose.Schema(
    {
        chatName: { type: String, trim: true },
        isGroupChat: { type: Boolean, default: false },
        users: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
        latestMessage: {
            type: mongoose.Schema.Types.ObjectId,
            ref: 'Message',
        },
        groupCreatedBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    },
    { timestamps: true }
);
```

Figure 77: Chat schema.

Based on the ERD of the message collection, I will create all fields with data types that match the ERD. In this collection, the sender and readBy fields will get the ID from the User schema to identify the sender and the person who read the message. The content field will be the content of the conversation and the chat field to identify the chat of the users.

```
const messageSchema = new mongoose.Schema(
    {
        sender: { type: mongoose.Schema.ObjectId, ref: 'User' },
        content: { type: String, trim: true },
        chat: { type: mongoose.Schema.ObjectId, ref: 'Chat' },
        readBy: [{ type: mongoose.Schema.ObjectId, ref: 'User' }],
    },
    { timestamps: true }
);
```

Figure 78: Message schema.

## 9.2. Front-end:

### 9.2.1. Project folder structure:

This is the folder structure on the front-end side of the project. Firstly, the node_modules folder will be the repository of the modules/libraries that I am using inside my project. When I do an npm install at that time, that module or library installs inside the node_module directory and an entry is added to the package.json file. Secondly, public folder will contains static files such as JavaScript library files, images, index.html, and other stuff, and so on that I do not want to handle with webpack. Next, all the files in this folder will be copied and pasted so they go straight to the build folder. References from HTML are only used by files inside the public folder. Thirdly, the src folder will contain the pages and logic, the project's images, the responsive folder, and the front-end application's libraries. The assets folder, which will contain the images and responsiveness for each user screen. The components folder will be the place to contain the product's pages and logic. The redux folder will be the place to store the data in the application based on the actions of the action folder to interact with the small JavaScript files of the components folder. The App.js file will be the parent component containing all the contents of the application.

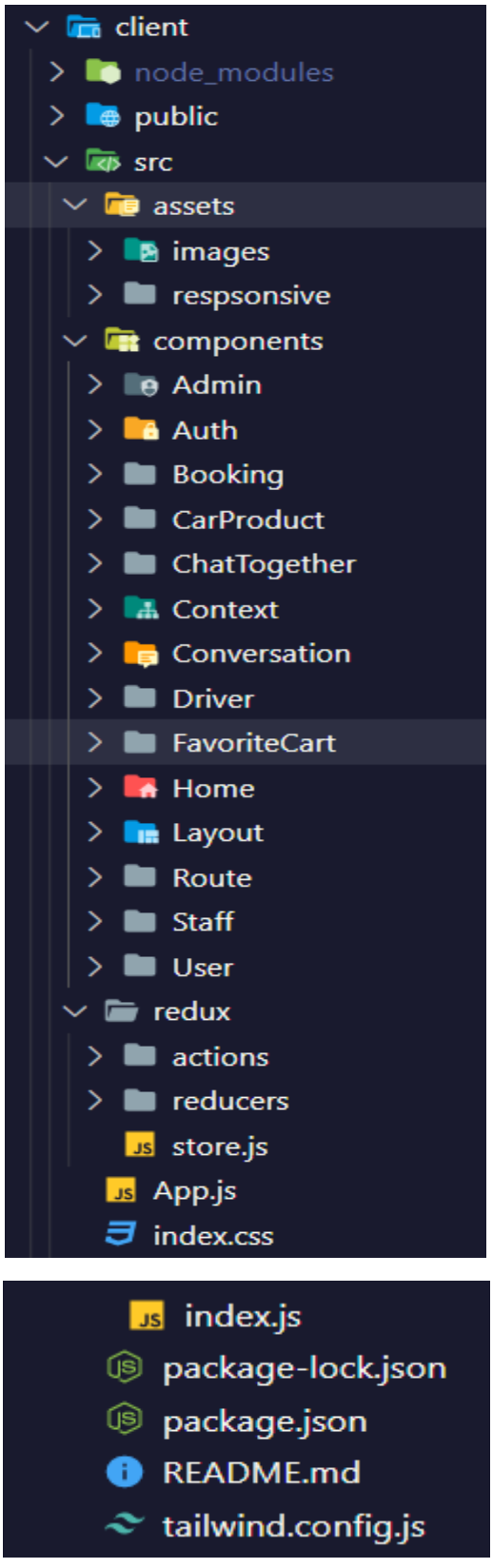



Figure 79: Folder structure front-end.

### 9.2.2. Source code samples:

**Code forgot password:**

For my application, I use the Yup library to validate forms easily. First, I need to create the initialValues variable to hold all the form input fields email, phoneNumber, and method. The input fields of the data to be validated by the validationSchema() function such as email and phone must be entered and the email must be the correct email in its formality. Next, I need to create forgotSubmit() function to pass data to forgotPassword method declared in redux action.

```
const { error, message, loading } = useSelector(
    (state) : any => state.forgotPassword
);

const initialValues = {
    email: '',
    phoneNumber: '',
    method: '',
};

const validationSchema = Yup.object({
    email: Yup.string()
        .email( message: 'Invalid email format !!')
        .required( msg: 'Please enter email !!'),
    phoneNumber: Yup.string().required( msg: 'Please enter phone !!'),
});

const forgotSubmit = (values) : void => {
    const { email, phoneNumber, method } = values;
    if (method === 'Phone') {
        navigate( to: '/forgotPassword/confirmOTP');
        dispatch(forgotPassword(email, phoneNumber, method));
    } else {
        dispatch(forgotPassword(email, phoneNumber, method));
    }
};

useEffect(() : void => {
    if (error) {
        toast.warn(error);
        dispatch(clearErrors());
    }
    if (message) {
        toast.warn(message.message);
    }
}, [dispatch, error, message]);
```

The function forgotPassword() is declared and takes 3 parameters email, phoneNumber and method. If unsuccessful, this function will report an error to the user. If successful, it will call forgotPassword api and pass 3 parameters email, phoneNumber and method to the server.

```
export const forgotPassword =
    (email, phoneNumber, method) : (dispatch: any) => Promise<... => async (dispatch) : Promise<void> => {
        try {
            dispatch({ type: 'FORGOT_PASSWORD_REQUEST' });

            const config = { headers: { 'Content-Type': 'application/json' } };

            const { data } = await axios.post(
                url: `/api/forgotPassword`,
                { email, phoneNumber, method },
                config
            );

            dispatch({ type: 'FORGOT_PASSWORD_SUCCESS', payload: data });
        } catch (error) {
            dispatch({
                type: 'FORGOT_PASSWORD_FAIL',
                payload: error.response.data.message,
            });
        }
    };
```

After calling the forgotPassword() function, the application will return a message and data to the component that uses the returned data.

```
export const forgotPasswordReducer = (state = {}, action) : {} => {
    switch (action.type) {
        case 'FORGOT_PASSWORD_REQUEST':
        case 'CONFIRM_OTP_REQUEST':
        case 'RESET_PASSWORD_REQUEST':
            return {
                ...state,
                loading: true,
                error: null,
            };
        case 'FORGOT_PASSWORD_SUCCESS':
            return {
                ...state,
                loading: false,
                message: action.payload,
            };
        case 'RESET_PASSWORD_SUCCESS':
            return {
                ...state,
                loading: false,
                success: action.payload,
            };
        case 'CONFIRM_OTP_SUCCESS': {
            return {
                ...state,
                loading: false,
                data: action.payload,
            };
        }
        case 'FORGOT_PASSWORD_FAIL':
        case 'CONFIRM_OTP_FAIL':
        case 'RESET_PASSWORD_FAIL':
            return {
                ...state,
                loading: false,
                error: action.payload,
            };

        case 'CLEAR_ERRORS':
            return {
                ...state,
                error: null,
            };

        default:
            return state;
    }
};
```

**Cole login with google:**

With the <GoogleLogin/> component there is a built-in onSuccess() function. And I will use this function to call the responseGoogle function that I declare.

```
<GoogleLogin
    clientId="1013536877025-ip5p8e6fhvjilej5ln9049isudh7s3k0.apps.googleusercontent.com"
    buttonText="Login with Google"
    onSuccess={responseGoogle}
    cookiePolicy={'single_host_origin'}
/>
```

The responseGoogle function receives the response parameter and the response parameter contains the ID of the real email account when the user activates with the login function with google and the ID variable will be declared with the tokenID variable to pass to the loginGoogle action declared in redux.

```
const responseGoogle = async (response) : Promise<void> => {
    const tokenId = response.tokenId;
    dispatch(loginGoogle(tokenId));
};
```

The loginGoogle() function will receive the tokenId variable. If the function fails, it will dispatch to “LOGIN_GG_FAIL” and return an error to the user. If the function executes successfully, it will call the googleLogin api and pass the tokenId variable to the server.

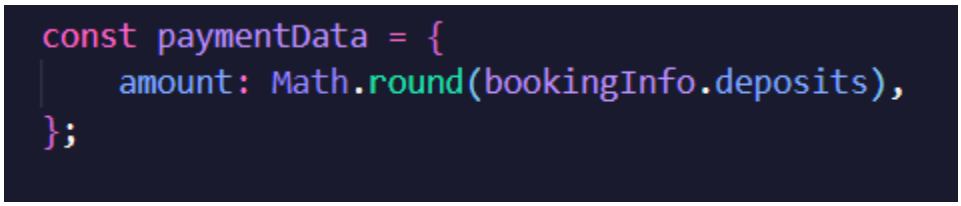
```
export const loginGoogle = (tokenId) : (dispatch: any) => Promise<... => async (dispatch) : Promise<void> => {
    try {
        dispatch({ type: 'LOGIN_GG_REQUEST' });
        const config = { headers: { 'Content-Type': 'application/json' } };

        const { data } = await axios.post( url: '/api/googleLogin', { tokenId }, config);

        dispatch({ type: 'LOGIN_GG_SUCCESS', payload: data });
    } catch (error) {
        dispatch({ type: 'LOGIN_GG_FAIL', payload: error.response.data.message });
    }
};
```

The loginGoogleReducer function to receive the values corresponding to the dispatch that the loginGoogle() action returns.

```
export const loginGoogleReducer = (state = initialState, action) : { user: {}; } | { loading: ... => {
    switch (action.type) {
        case 'LOGIN_GG_REQUEST':
        case 'LOGIN_FB_REQUEST':
            return {
                loading: true,
                isLoggedIn: false,
            };

        case 'LOGIN_GG_SUCCESS':
        case 'LOGIN_FB_SUCCESS':
            return {
                ...state,
                loading: false,
                isLoggedIn: true,
                message: action.payload.message,
                token: action.payload.token,
                user: action.payload.user,
            };

        case 'LOGIN_GG_FAIL':
        case 'LOGIN_FB_SUCCESS':
            return {
                ...state,
                loading: false,
                isLoggedIn: false,
                user: null,
                error: action.payload,
            };
        case 'CLEAR_ERRORS':
            return {
                ...state,
                error: null,
            };

        default:
            return state;
    }
};
```

**Code payment with stripe:**

This is the online payment function with Stripe. First, I need to declare the variable paymentData and the variable paymentData as the amount that the customer has to pay. Next, I declare the submitHanlder() function to perform the payment function. If the function fails, it will go into the catch function and display the error to the user. If the function works successfully, it will call the paymentStripe api and pass the data (paymentData) to the server. Then, I declare the client_secret variable to get the data.client_secret data from the server returned. Then, I call the result function to pass the user's information (credit card number, name, email and address) to Stripe. Next, if Stripe cannot accept the credit card from the user transaction, the system will give an error. Conversely, if Stripe receives a credit card from a user transaction, it will call createBooking() and this function will let the application collect all the information the user has booked. After successfully booking the car, the user will be redirected to the successful booking page.



```
const submitHandler = async (e) : Promise<void> => {
    e.preventDefault();

    payBtn.current.disabled = true;

    try {
        const config = {
            headers: {
                'Content-Type': 'application/json',
            },
        };
        const { data } = await axios.post(
            url: '/api/booking/paymentStripe',
            paymentData,
            config
        );

        const client_secret = data.client_secret;

        if (!stripe || !elements) return;

        const result = await stripe.confirmCardPayment(client_secret, {
            payment_method: {
                card: elements.getElement(CardNumberElement),
                billing_details: {
                    name: user.name,
                    email: user.email,
                    address: {
                        line1: receivingCarTo.address,
                        city: receivingCarTo.location,
                    },
                },
            },
        });

        if (result.error) {
            payBtn.current.disabled = false;

            toast.error(result.error.message);
        } else {
            if (result.paymentIntent.status === 'succeeded') {
                book.paymentInfo = {
                    id: result.paymentIntent.id,
                    status: result.paymentIntent.status,
                };
                dispatch(createBooking(book));
                navigate(to: '/paymentSuccess');
            } else {
                toast.error(content: 'There is some issue while processing payment !!');
            }
        }
    } catch (error) {
        payBtn.current.disabled = false;
        toast.error(error.response.data.message);
    }
};
```

**Code chatbot:**

First, I declare the userSendFrequentlyAskEvent() function so that the application can automatically notify the user about what the user needs the chatbot to help with. The userSendFrequentlyAskEvent() function receives the event variable when calling the function. If the function encounters an error, it will run into the catch function and the chatbot will display an error message to the user "Error, please check the problems!!". If the function doesn't have a problem, the function calls the sendFrequentlyAskToChatBot api and passes the event variable to the server. Then I create a conversation variable to receive data from the server and pass the conversation variable to the saveMessage() function.

```
const userSendFrequentlyAskEvent = async (event) : Promise<void> => {
    try {
        const config = {
            headers: { 'Content-Type': 'application/json' },
        };

        const { data } = await axios.post(
            url: '/api/chatbot/sendFrequentlyAskToChatBot',
            {
                event,
            },
            config
        );

        let conversation = {
            who: 'ChatBot',
            content: data.chatBotResponseFrequentlyAskEvent,
        };
        dispatch(saveMessage(conversation));
    } catch (error) {
        let conversation = {
            who: 'ChatBot',
            content: {
                text: {
                    text: ' Error, please check the problem',
                },
            },
        };
        dispatch(saveMessage(conversation));
    }
};
```

The saveMessage() function is declared to payload data and return data in chatBotReducer for the purpose of displaying the chatbot's conversation to the user.

```
export function saveMessage(dataToSubmit) : { type: string; payload: an... {
    return {
        type: 'SAVE_MESSAGE',
        payload: dataToSubmit,
    };
}
```

```
export const chatBotReducer = (state = { messages: [] }, action) : { messages: any[]; } => {
    switch (action.type) {
        case 'SAVE_MESSAGE':
            return {
                ...state,
                messages: state.messages.concat(action.payload),
            };
        default:
            return state;
    }
};
```

Finally, I use the useEffect function to call the userSendFrequentlyAskEvent() function and pass the variable 'WelcomeToTCAR' after each time the user enters the Home page.

```
useEffect(() : void => {
    userSendFrequentlyAskEvent(event: 'WelcomeToTCAR');
}, []);
```

Next, I declare the userSendMessage() function and this function is intended to receive questions from users and the chatbot will automatically answer their questions. If the function encounters an error, it will run into the catch function and the chatbot will display an error message to the user "Error, please check the problems!!". If this function does not have problems, it will call the sendMessageToChatBot api and turn the text variable to the server. The conversation variable will receive the data returned from the server and pass it to the saveMessage() function.

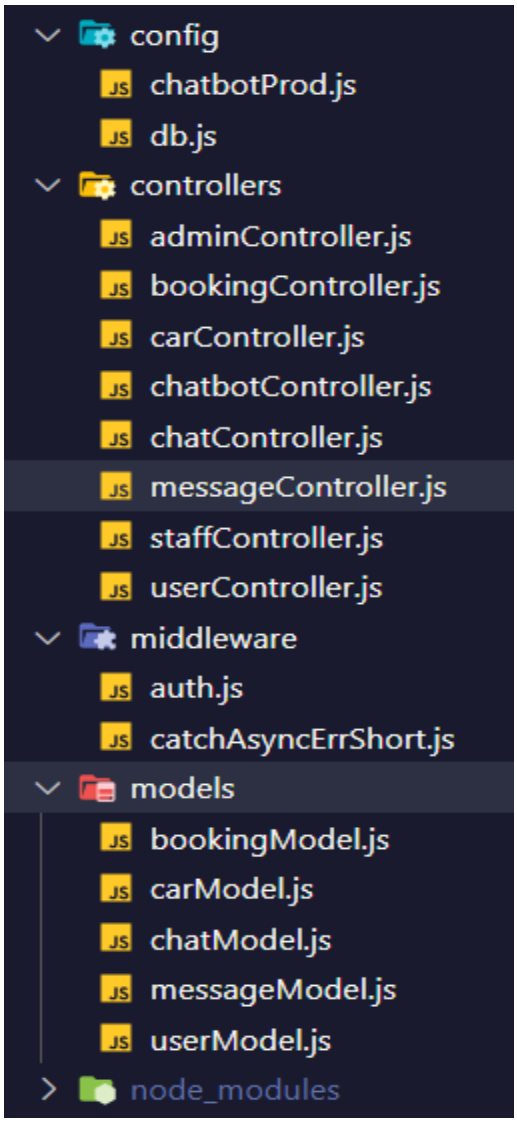
```
const userSendMessage = async (text) : Promise<void> => {
    //  Yser send message
    let conversation = {
        who: 'User',
        content: {
            text: {
                text: text,
            },
        },
    };

    dispatch(saveMessage(conversation));

    try {
        const config = {
            headers: { 'Content-Type': 'application/json' },
        };
        const { data } = await axios.post(
            url: '/api/chatbot/sendMessageToChatBot',
            {
                text,
            },
            config
        );
        conversation = { who: 'ChatBot', content: data.chatBotResponse };

        dispatch(saveMessage(conversation));
    } catch (error) {
        conversation = {
            who: 'ChatBot',
            content: {
                text: {
                    text: 'Error, please check problems !!',
                },
            },
        };
        dispatch(saveMessage(conversation));
    }
};
```

## 9.3. Back-end:

### 9.3.1. Project folder structure:

This is the directory structure on the back-end of the project. Firstly, the config folder will contain the configurations of the function that connects to the database and the chatbot. Secondly, the folder controller is used to handle logical thoughts and data when NodeJS works with the database. Thirdly, folder middleware is seen as a standing guard that prevents errors

or improper requests when processing functions in folder controllers. Fourth, the models folder will be the place to design and store the data fields of the database. Fifthly, the node_modules folder will be the repository of the modules/libraries that I am using inside my project. When I do an npm install at that point, that module or library installs inside the node_module folder and an entry is added to the package.json file. Sixthly, the routes folder will be the place to configure the functions from the controllers folder with the http method. Seventhly, the utils folder will contain the JavaScript files that support the functions in the controller folder such as data retrieval, send email, what token the user logged in with, and so on. Eighthly, the config.env file will be the place to contain information that I do not want to display on GitHub such as the link to the database on mongoCloud, the chatbot's security information, and so on. Finally, the server.js file will be the file containing all https requests to send the API to the front-end and return the data that the front-end sends to the database.



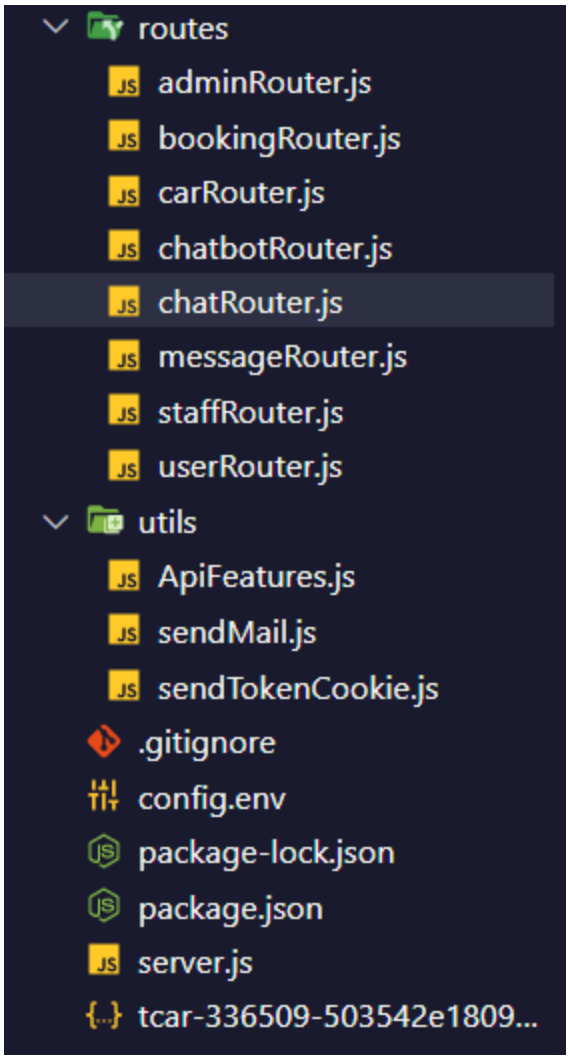



Figure 80: Folder structure back-end.

**9.3.2. Source code samples:**

**Code forgot password with email or SMS (phone):**

This is the code forgot password of the backend. First, I declare 3 variables email, method, phoneNumber that the server will receive from the front-end. Second, I need to check if the user account exists in the system? Third, I declare 2 variables resetPWToken and OTP. resetPWToken for user to change password with email and OTP for user to change password with SMS. Next, if the user chooses the method as email, they will receive an email sent back from the server with the password reset link. Otherwise, if the user chooses the method with the Phone, they will receive an SMS with the OTP code for the user for them to use change password.

```
exports.forgotPassword = async (req, res) : Promise<any> => {
    try {
        const { email, method, phoneNumber } = req.body;
        const user = await User.findOne({ email });
        if (!user) {
            return res.status(404).json({ message: 'User not found !!' });
        }

        const resetPWToken = user.getResetPasswordToken();
        const OTP = user.getResetPassWordOTP();

        const resetPasswordUrl = `${CLIENT_URL_DEPLOY}/resetPassword/${resetPWToken}`;
        // CLIENT_URL_DEPLOY;
        await user.save({ validateBeforeSave: false });

        if (method === 'Email') {
            try {
                await sendEmail(
                    email,
                    resetPasswordUrl,
                    message: 'Please click to reset your password !!'
                );

                res.status(200).json({
                    message: 'Please check your email to reset your password !!',
                });
            } catch (err) {
                user.resetPasswordToken = undefined;
                user.resetPasswordExpireIn = undefined;
                await user.save({ validateBeforeSave: false });
                return res.status(500).json({ message: err.message });
            }
        } else if (method === 'Phone') {
            try {
                const accountSid = process.env.TWILIO_ACC_SID;
                const authToken = process.env.TWILIO_AUTH_TOKEN;
                const client = require(id: 'twilio')(accountSid, authToken);    2.3M (gzipped: 202.6k)

                const sendOTP = await client.messages.create({
                    body: `${OTP}`,
                    from: '+19706717801',
                    to: phoneNumber,
                    // to: '+84905092786',
                });

                res.status(200).json({
                    email,
                    sendOTP,
                    message: 'Please check your phone to reset your password !!',
                });
            } catch (err) {
                user.resetPasswordToken = undefined;
                user.resetPasswordExpireIn = undefined;
                await user.save({ validateBeforeSave: false });
                return res.status(500).json({ message: err.message });
            }
        }
    } catch (err) {
        return res.status(500).json({ message: err.message });
    }
};
```

**Cole login with google:**

This is the backend's login code with Google. First, I declare the tokenId variable that the server will receive from the front-end. Second, I need the verifyIdToken that the front-end returns. After the tokenId variable is verified, I will set it with the dataVerify variable. Next, I'll get the email, name, picture, sub, email_verified fields from the user's email coming in from the front-end. Next, I need to hash the user's password. Next, I check if the email the user logged in to is the real email? If it is a real email, user will be able to login successfully. In the login section, if the user already has an account that exists in the database, then they just need to login with the token cookie. Conversely, if the user does not have an account in the database, they will proceed to add their account to the database.

```
exports.googleLogin = async (req, res) : Promise<any> => {
    try {
        const { tokenId } = req.body;

        const dataVerify = await client.verifyIdToken({
            idToken: tokenId,
            audience: process.env.MAILING_SERVICE_CLIENT_ID,
        });

        const { email_verified, email, name, picture, sub } = dataVerify.payload;
        const password = email + process.env.GOOGLE_SECRET;
        const hashPassword = await bcrypt.hash(password, saltOrRounds: 12);

        if (!email_verified) {
            return res.status(400).json({ message: 'This email is not exist !!' });
        }

        const user = await User.findOne({ email });

        if (user) {
            sendTokenCookie(user, responseStatusCode: 200, res, text: 'Login success with google !!');
        } else {
            await cloudinary.v2.uploader.upload(picture, {
                folder: 'avatars',
                width: 150,
                crop: 'scale',
            });
            const newUser = new User({
                name,
                email,
                password: hashPassword,
                avatar: {
                    public_id: sub,
                    url: picture,
                },
            });
            await newUser.save();
            sendTokenCookie(newUser, responseStatusCode: 200, res, text: 'Login success with google !!');
        }
    } catch (err) {
        return res.status(500).json({ message: err.message });
    }
};
```

**Code payment with stripe:**

Here is the payment code with Stripe from the back end. The back end only needs to receive the data (amount) from the back-end to create the payment function. This function will return the client_secret variable so that the front-end gets the value that the front-end makes the request.

```
exports.paymentStripe = catchAsyncErrShort(async (req, res) : Promise<void> => {
    const payment = await stripe.paymentIntents.create({
        amount: req.body.amount,
        currency: 'inr',
        metadata: {
            company: 'Ecommerce',
        },
    });

    res.status(200).json({
        message: 'Payment with Stripe success !!',
        client_secret: payment.client_secret,
    });
});
```

**Code manage booking:**

First, I need to declare the updateStatusBooking() function so that the application automatically updates the vehicle status when the user has booked. The function takes 4 values id, now, startDayBook, endDayBook, carId. I declare the variable book to find booking by id and car to find car by id. If the current time is less than the start date of the car rental, the booking is in Processing status and the car has been booked. If the current time is greater than or equal to the car rental start date and less than or equal to the car rental end date, the booking is in IsRunning status and the car has been booked. If the current time is greater than the rental end date, the booking is in Done status and the car needs to be updated with the rental date.

```
async function updateStatusBooking(id, now, startDayBook, endDayBook, carID) : Promise<void> {
    const book = await Booking.findById(id);
    const car = await Car.findById(carID);

    if (now < startDayBook) {
        book.bookingStatus = 'Processing';
        car.available = 'isBooked';
    }

    if (now >= startDayBook && now <= endDayBook) {
        book.bookingStatus = 'isRunning';
        car.available = 'isBooked';
    }

    if (now > endDayBook) {
        book.bookingStatus = 'Done';
        car.available = 'Update';
    }

    await book.save({ validateBeforeSave: false });
    await car.save({ validateBeforeSave: false });
}
```

Next, I declare the getAllBooking function so that the admin or staff can manage all the bookings of the system. This function will display filtered data with pagination of 5 and calculate the total amount of trips. The function will call the map() method to call the updateStatusBooking() function to get the Id of each booking from the bookings object of the database for the purpose of always updating the bookings with the status of that booking in real time.

```
exports.getAllBooking = catchAsyncErrShort(async (req, res) : Promise<void> => {
    const resultItemPage = 5;
    const booksCount = await Booking.countDocuments();
    const apiFeature = new ApiFeatures(Booking.find(), req.query)
        .pagination(resultItemPage)
        .sort();
    const books = await apiFeature.query;

    let totalAllPrice = 0;

    const booking = await Booking.find();

    booking.forEach((book) : void => {
        totalAllPrice += book.totalPrice;
    });

    booking.map(async (book) : Promise<void> => {
        const id = book.id;
        const carID = book.bookCars[0].car;
        const now = moment(moment().startOf(unitOfTime: 'hour'));
        const startDayBook = moment(
            moment(book.bookCars[0].startDay).startOf(unitOfTime: 'hour')
        );
        const endDayBook = moment(moment(book.bookCars[0].endDay).startOf(unitOfTime: 'hour'));
        await updateStatusBooking(id, now, startDayBook, endDayBook, carID);
    });

    res
        .status(200)
        .json({ success: true, totalAllPrice, booksCount, resultItemPage, books });
});
```

## 9.4. Deployment:

First, I need to set up a Heroku account and push my code to GitHub.

**Link GitHub:** https://github.com/DinhMinh2810/TCAR_App_Project

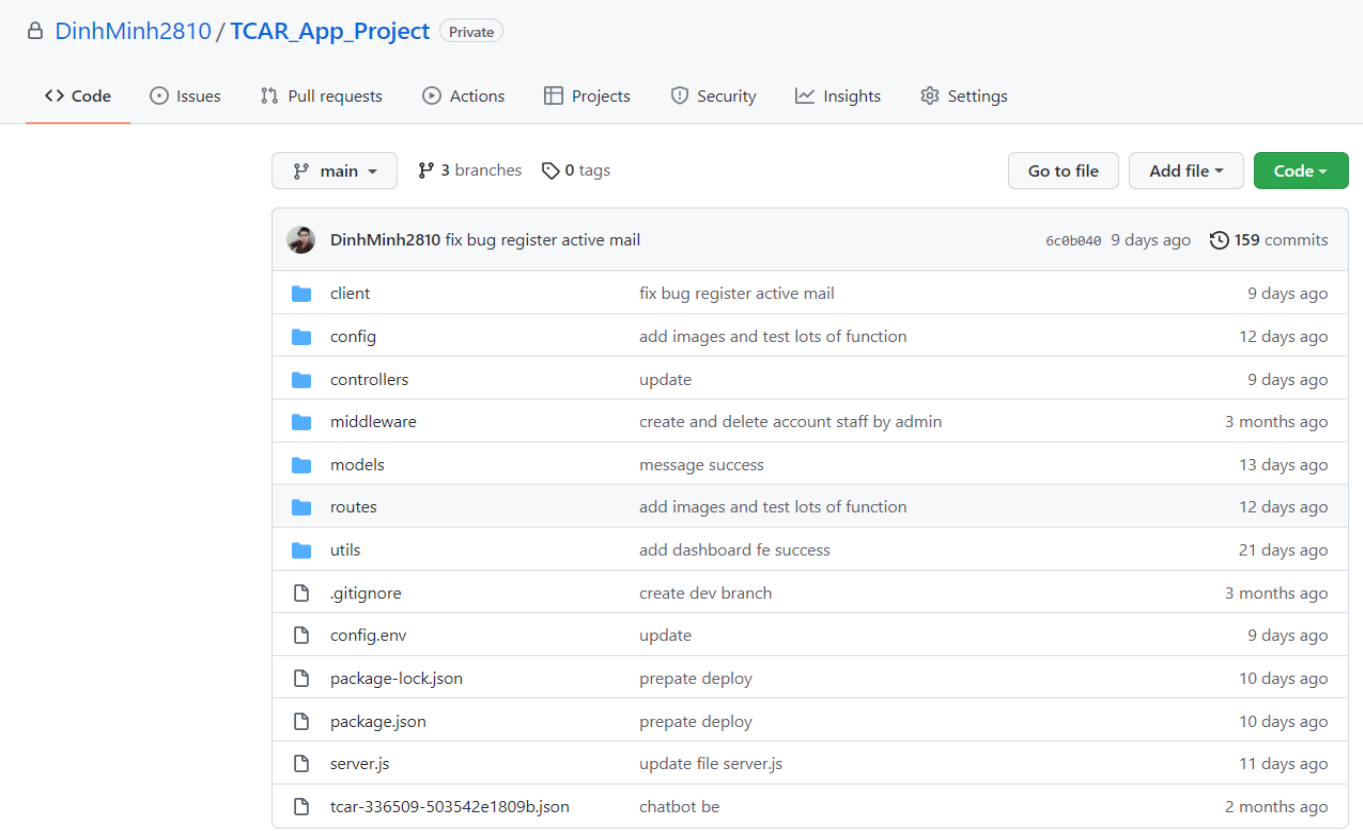



Figure 81: Github of TCAR.

Then, I need to implement requests based on a few lines of code in the server.js file of the backend. If the value in the development variable is production, then execute the build file index.html in the client folder.

```
// Deploy
if (process.env.NODE_ENV === 'production') {
    app.use(express.static('client/build'));
    app.get('*', (req, res) => {
        res.sendFile(path.join(__dirname, 'client', 'build', 'index.html'));
    });
}
```

```
"scripts": {
  "start": "node server.js",
  "server": "nodemon server.js",
  "heroku-postbuild": "cd client && npm install && npm run build"
},
```

Next, I need to set the environment variables in the config.env file on Heroku.

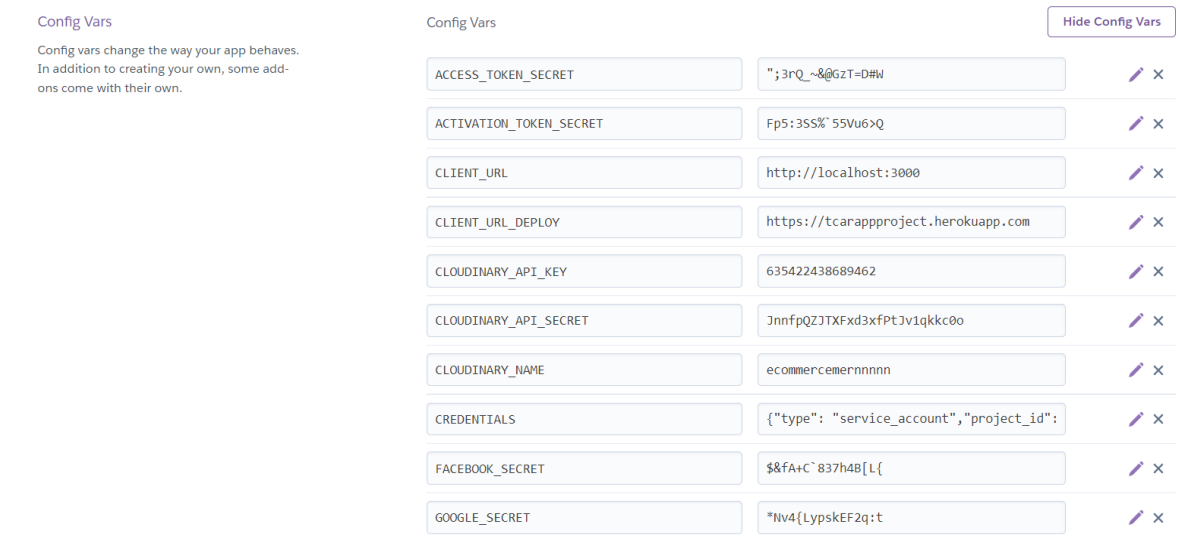



Figure 82: Config variables in Heroku

Next, I connect my GitHub as the deploy method on Heroku. I use the command “heroku git: remote -a tcarapproject” to find my GitHub link to test the connection with the Heroku app. After confirming that my GitHub was connected to Heroku, I proceeded to use the command “git push Heroku main” to deploy my application with branch main.

```
PS C:\TCAR_App> heroku git:remote -a tcarappproject
 »   Warning: heroku update available from 7.53.0 to 7.60.1.
set git remote heroku to https://git.heroku.com/tcarappproject.git
PS C:\TCAR_App> git remote -v
heroku  https://git.heroku.com/tcarappproject.git (fetch)
heroku  https://git.heroku.com/tcarappproject.git (push)
origin  git@github.com:DinhMinh2810/TCAR_App_Project.git (fetch)
origin  git@github.com:DinhMinh2810/TCAR_App_Project.git (push)
PS C:\TCAR_App> git push heroku main
Enumerating objects: 2722, done.
Counting objects: 100% (2722/2722), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2600/2600), done.
Writing objects: 100% (2722/2722), 2.39 MiB | 500.00 KiB/s, done.
Total 2722 (delta 1695), reused 0 (delta 0), pack-reused 0
```

Figure 83: Deploy.

Deployment takes about 3-5 minutes depending on the network. And great, my terminal says that it has been deployed successfully.

```
remote:
remote:  !
remote:  ! ## Warning - The same version of this code has already been built: 6c0b040715cfd009071a2c720f4211d65edfe15a
remote:  !
remote:  ! We have detected that you have triggered a build from source code with version 6c0b040715cfd009071a2c720f4211d65edfe15a
remote:  ! at least twice. One common cause of this behavior is attempting to deploy code from a different branch.
remote:  !
remote:  ! If you are developing on a branch and deploying via git you must run:
remote:  !
remote:  !     git push heroku <branchname>:main
remote:  !
remote:  ! This article goes into details on the behavior:
remote:  !   https://devcenter.heroku.com/articles/duplicate-build-version
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/tcarappproject.git
 * [new branch]      main -> main
```

Figure 84: Deploy.

Finally, I go to the Heroku app and open the TCAR app.

**Link deploy:** https://tcarappproject.herokuapp.com/

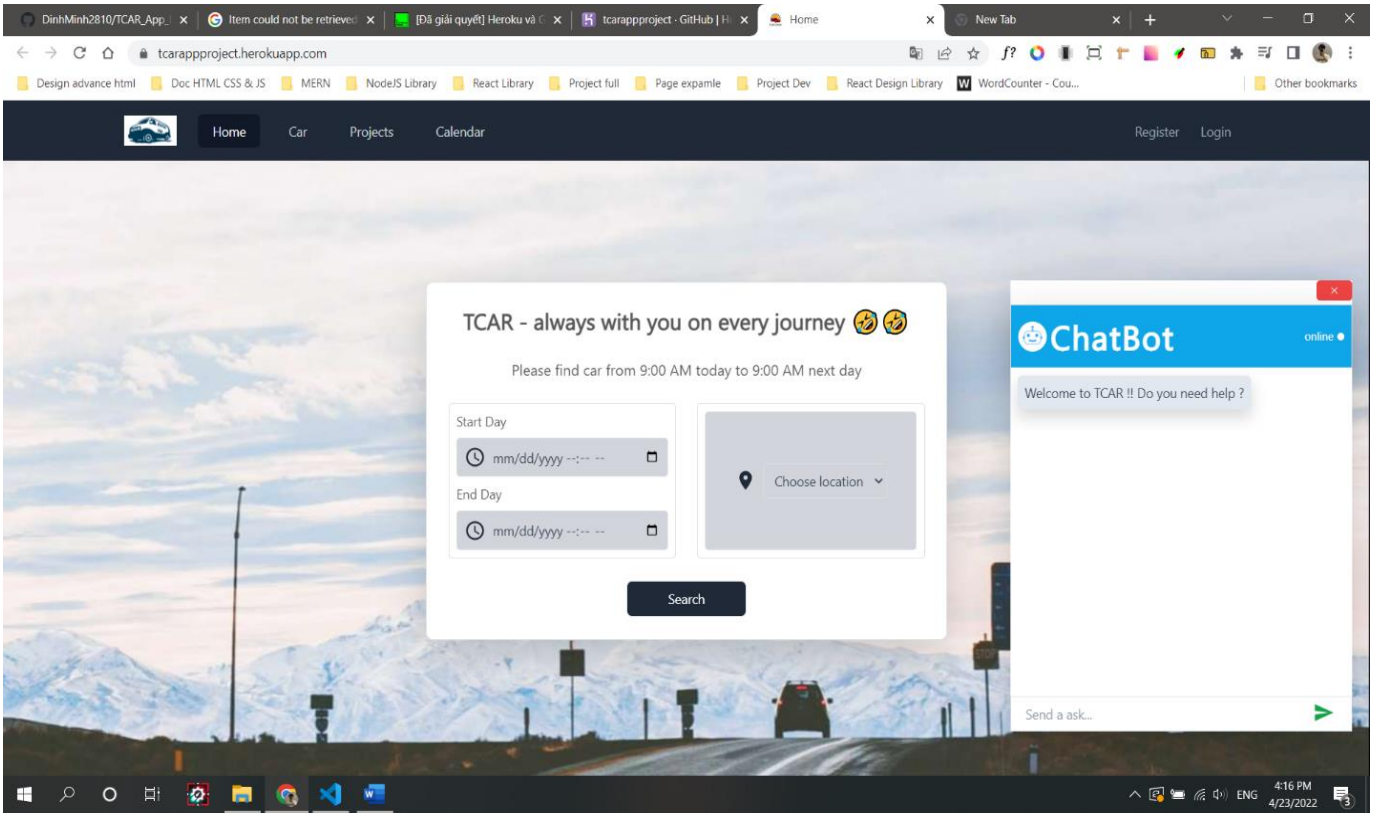



Figure 85: Deploy success.

## 9.5. Images (Demo of car rental process):

First, the admin will create a car with complete information fields to display to users.

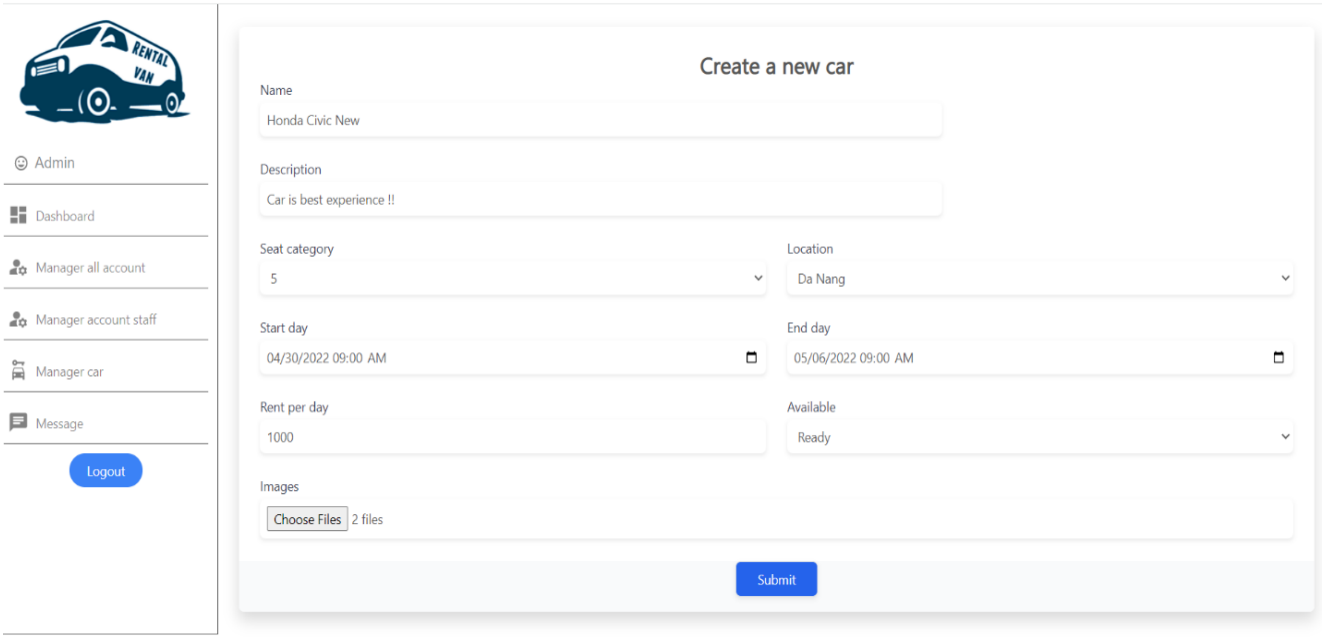



Figure 86: Create a new car page.

Next, the staff will create an account for the driver that corresponds to the location where the new car has just been created by the admin.

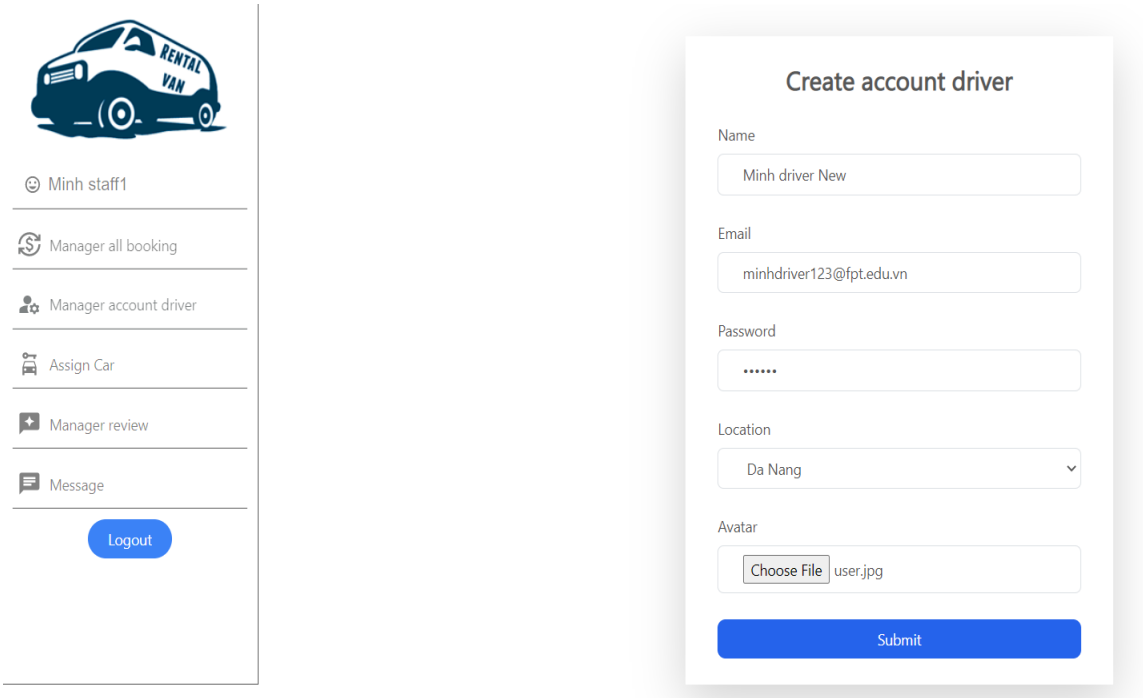



Figure 87: Create account driver.

Next, the staff will assign all cars to the driver that requires the same location or that driver has not been assigned to another car.

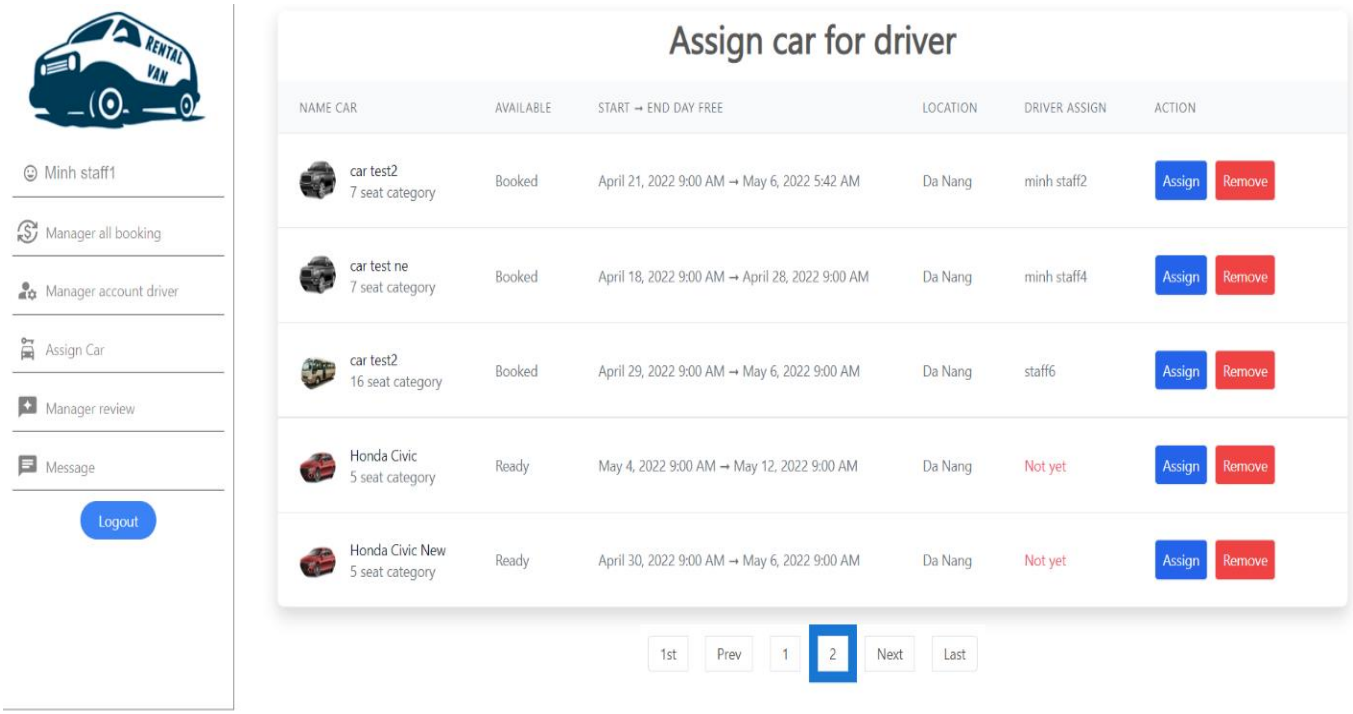



Figure 88: List car assign or not assign.

This is a list of all available drivers to assign to the vehicle the staff wants to assign for driver.

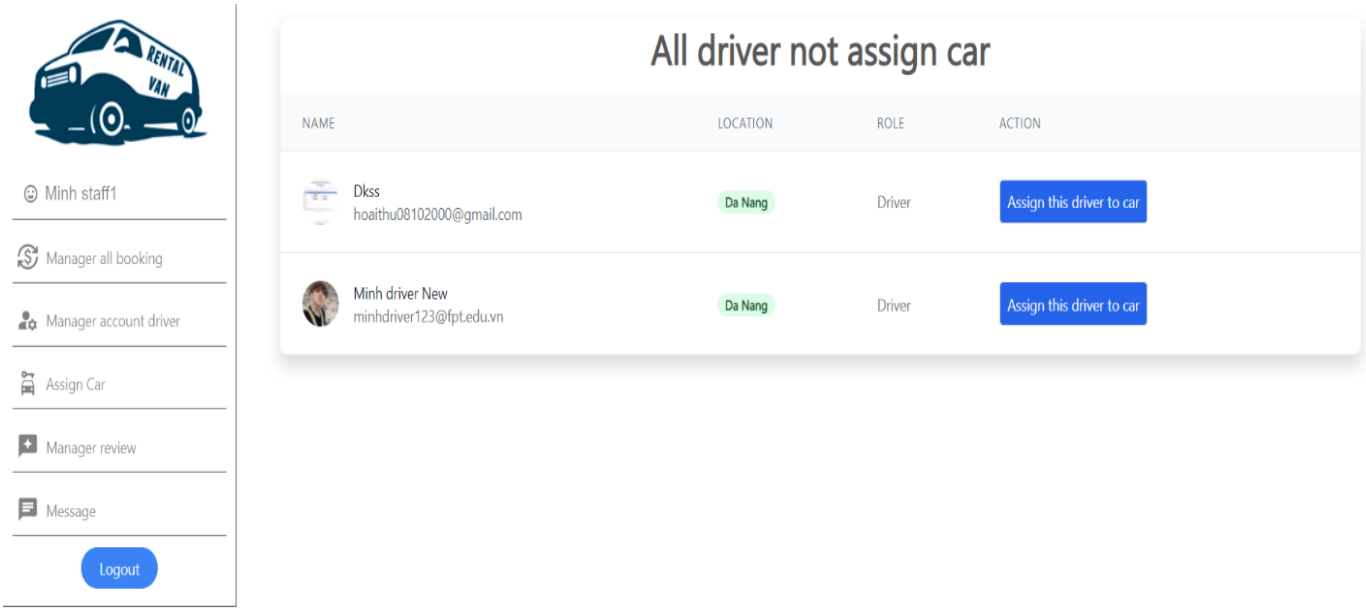



Figure 89: List driver not assign car.

Next, as a user, the user must log into the system. The system has 3 ways to log in to the system which is login with account and password or login with google or login with Facebook. However, here I only let the user login with google.

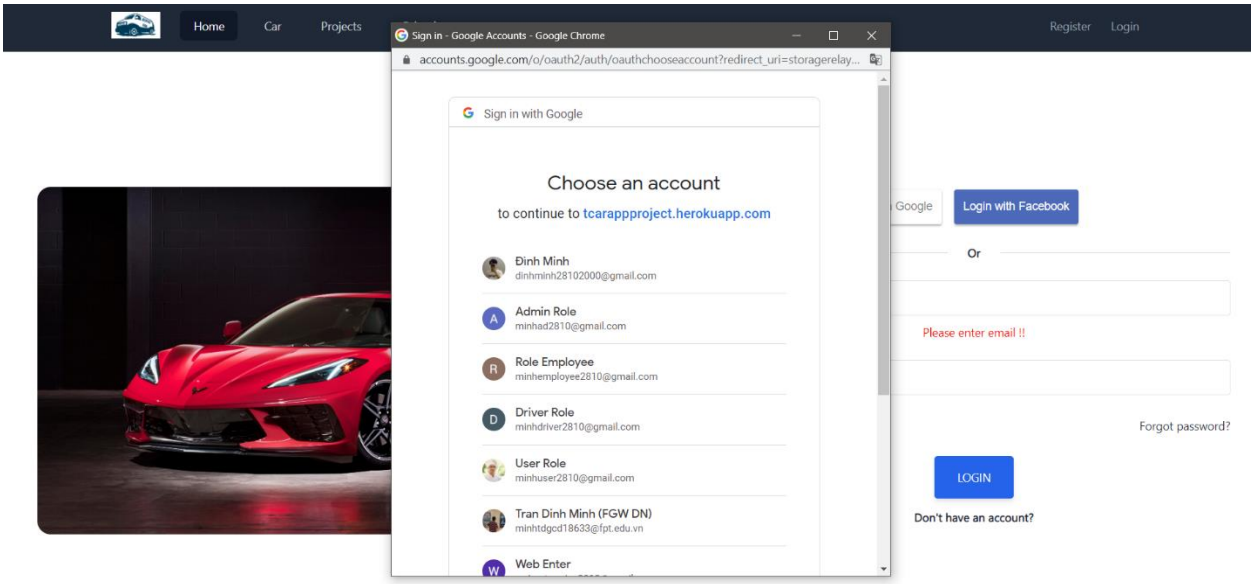



Figure 90: Login with google.

After successfully logging into the system, the user will go to the home page. Users can search for cars in 3 ways, click the car button to see all cars, or search for cars by start and end date when the user wants to find a specific car, or the user is supported by the chatbot to find a car.

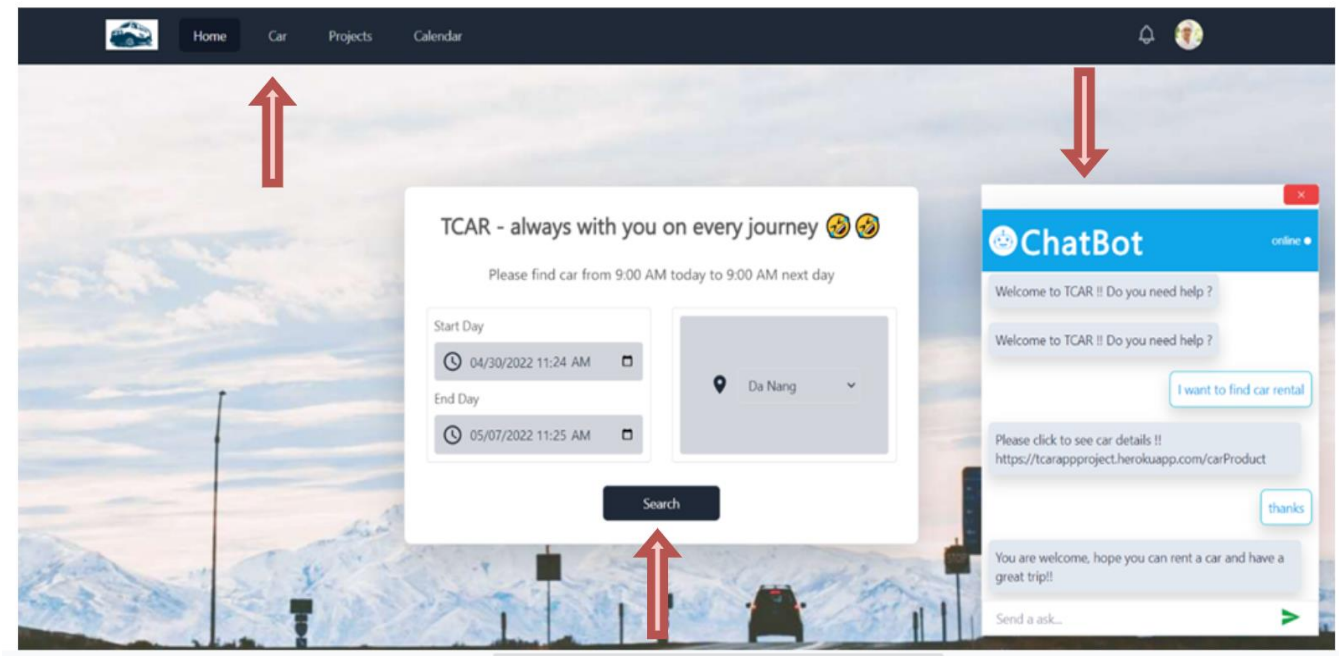



Figure 91: Find car.

Next, users will see all the cars they want to rent, if they want to see the car details they click on that car.

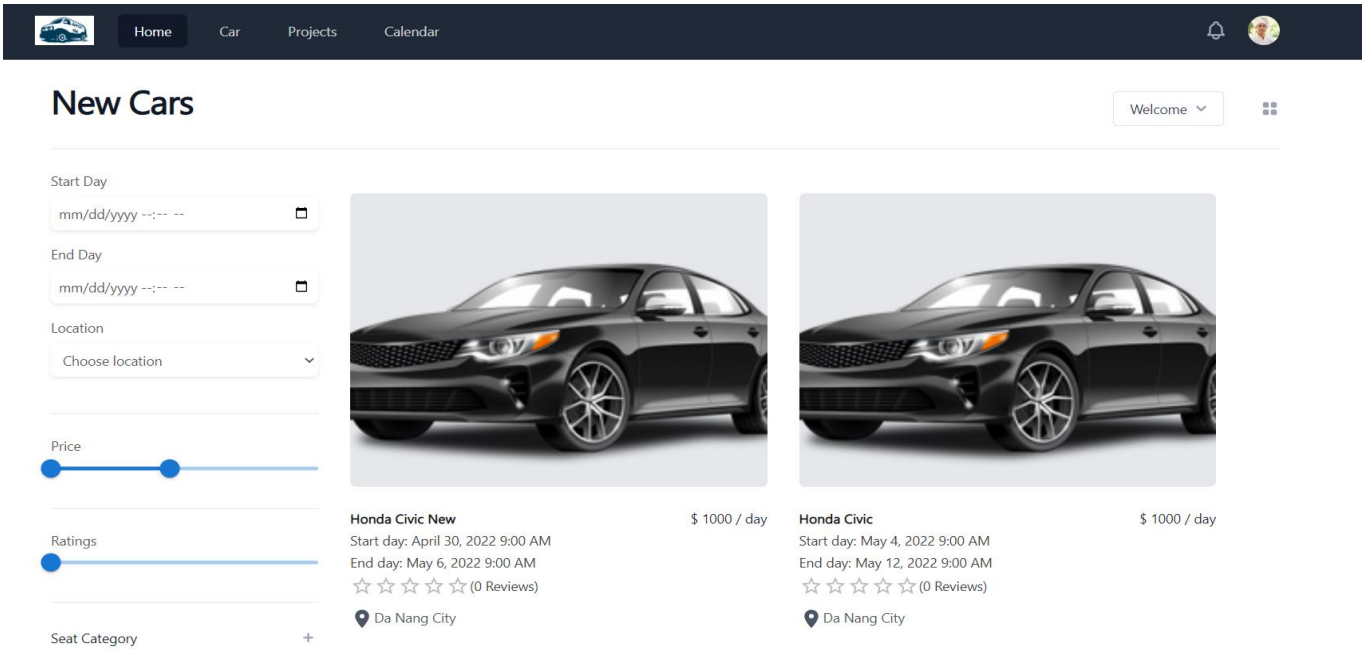



Figure 92: All car page.

The user will choose a rental date, the car can only be rented with the time from the start date of the car, and the user can rent the car for as long as he wants. After, user click button “Book car right now” to rental car.

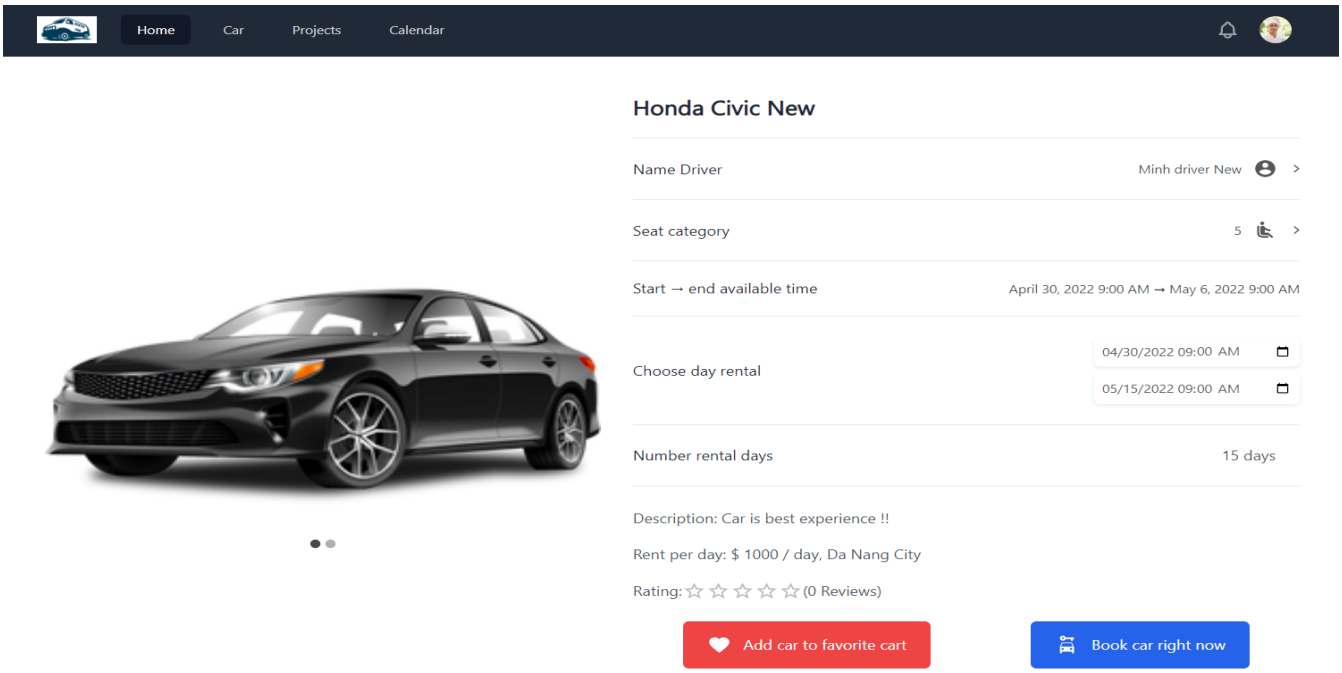



Figure 93: Car detail page.

Next, users need to enter their necessary information to be able to rent a car.

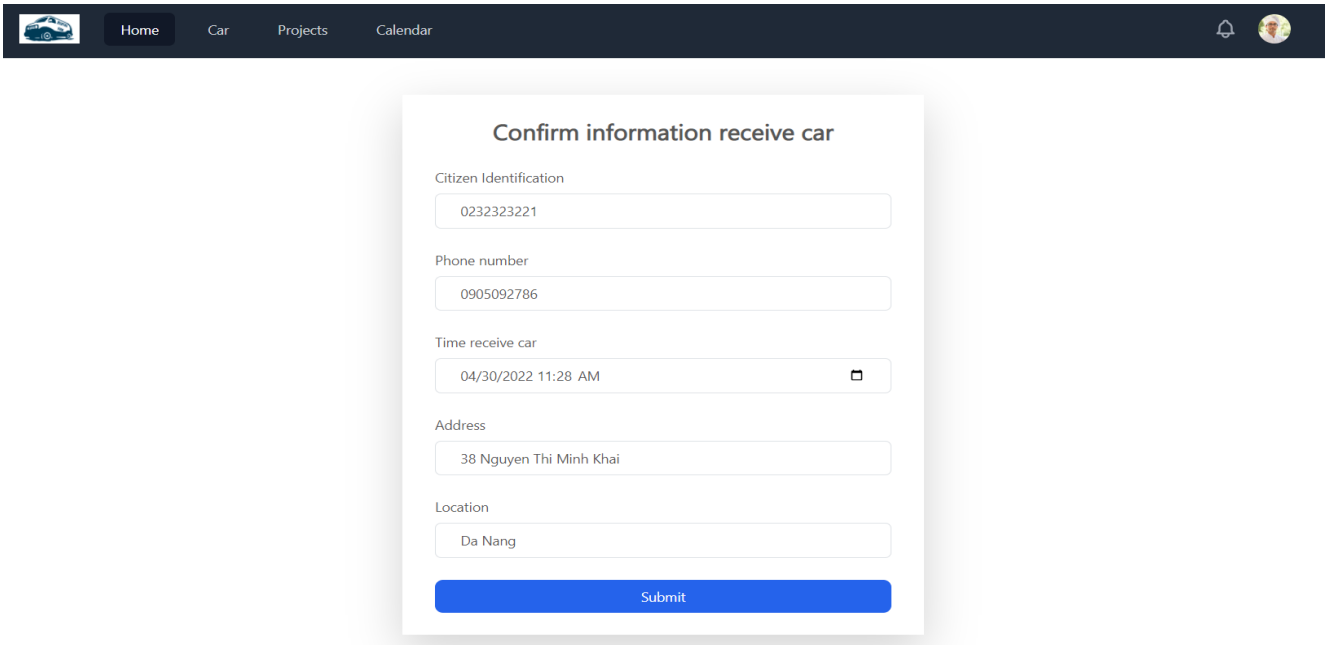



Figure 94: Confirm information receive car page.

Next, users will be able to check their car rental information, car rental date and information their and then can pay in 2 forms Stripe or Braintree.

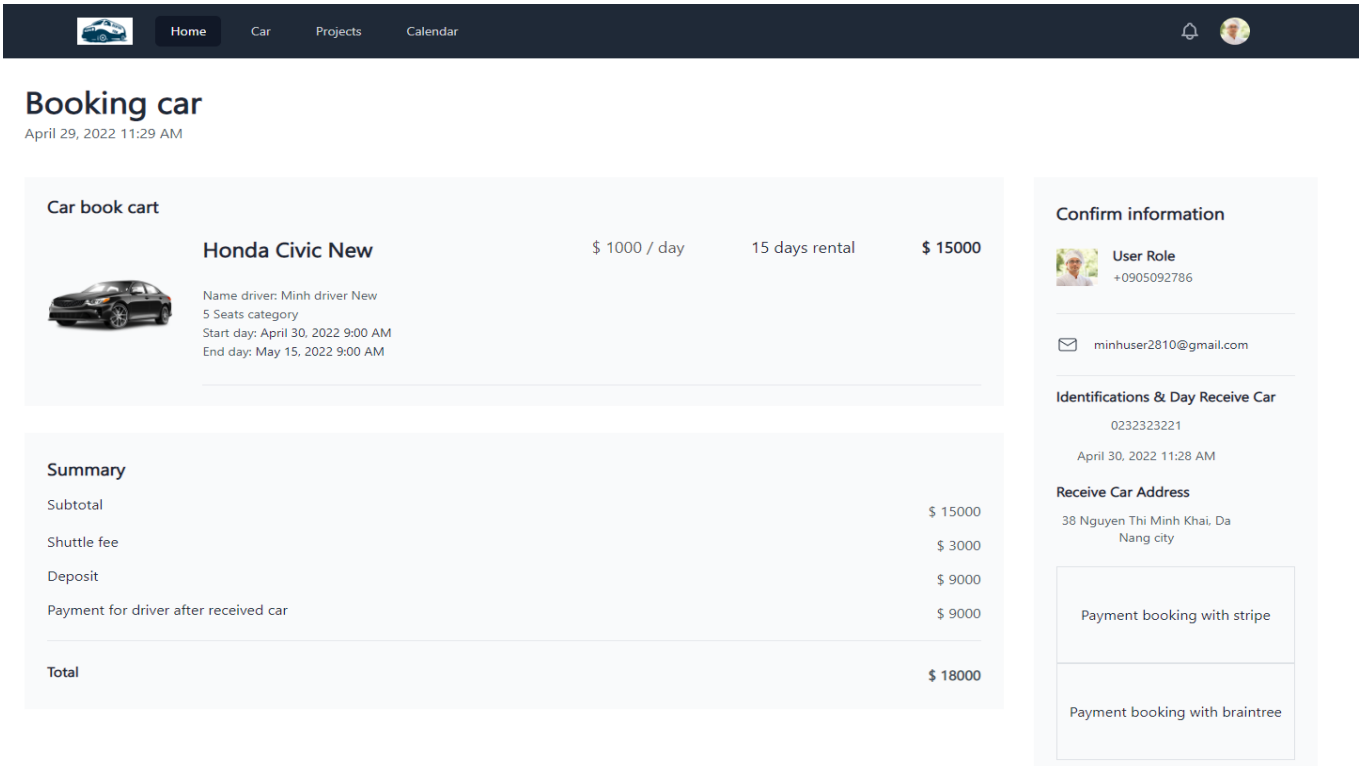



Figure 95: Confirm information booking page.

For this demo, I let users pay with Stipe, users will need to enter their credit card information to pay.

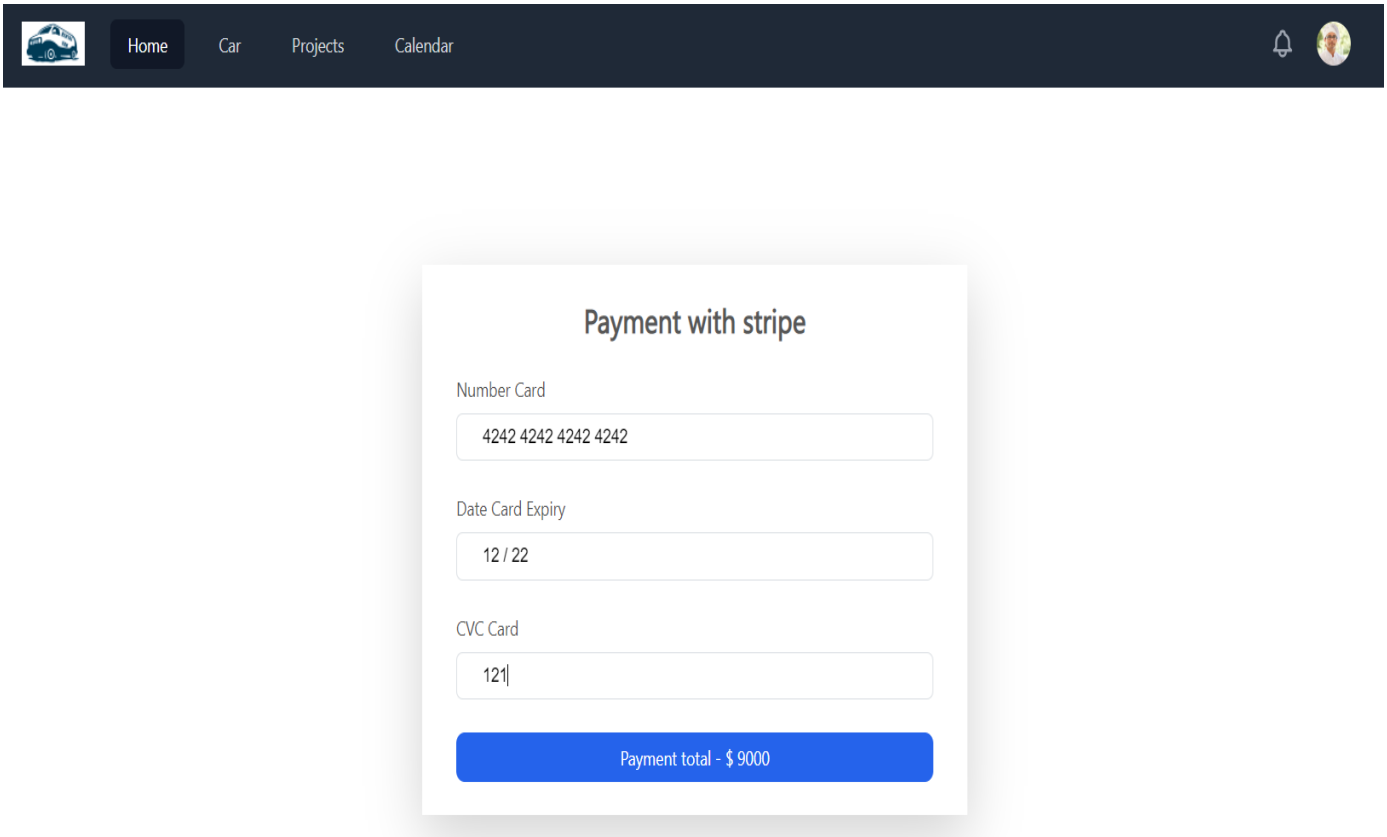



Figure 96: Payment with Stripe page.

After the user has successfully paid, the system will go to the page where the user has successfully booked the car.

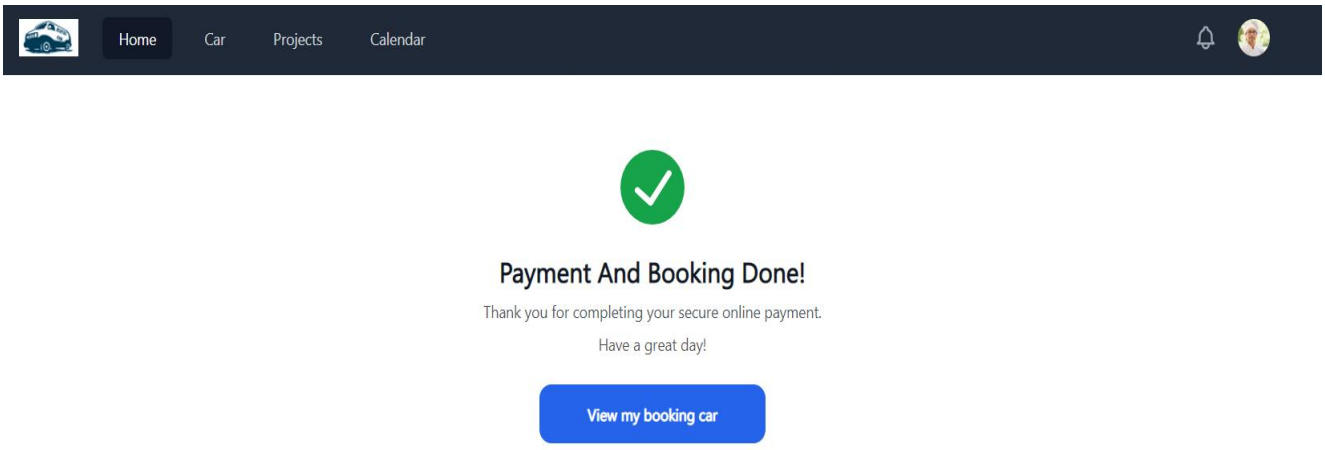



Figure 97: View booking success page.

Next, the user can review the ride they have booked. They can even click the “See detail” button to view their booking details.

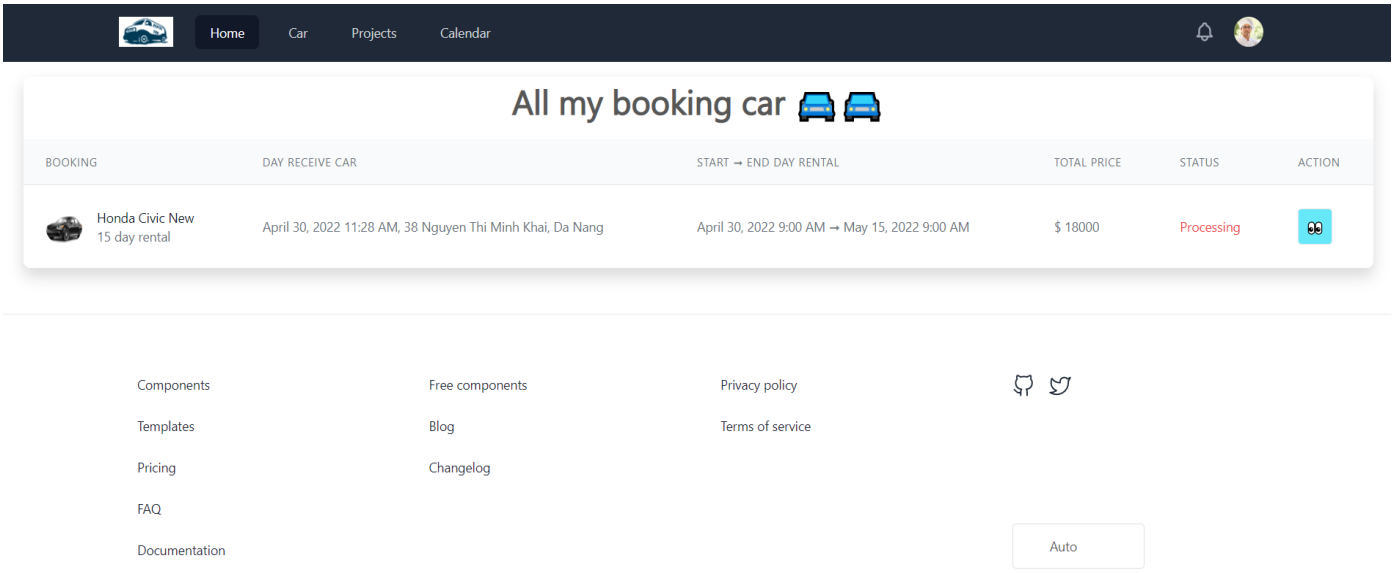



Figure 98: View all user booking.

Finally, users see the details of the trip they have booked.

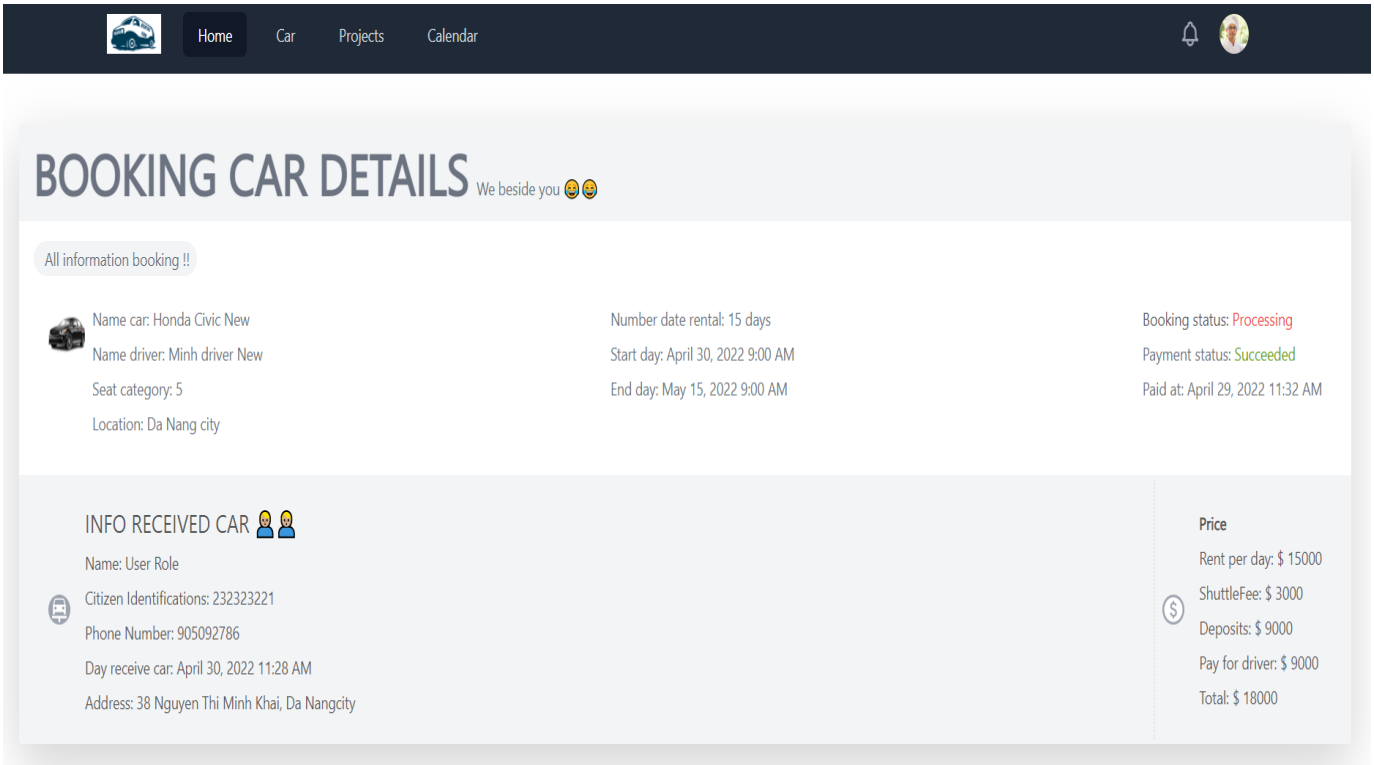
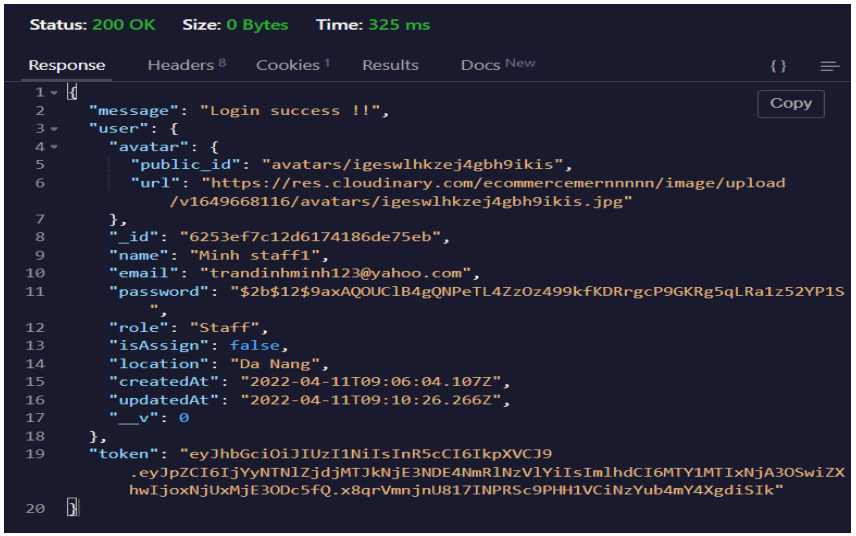
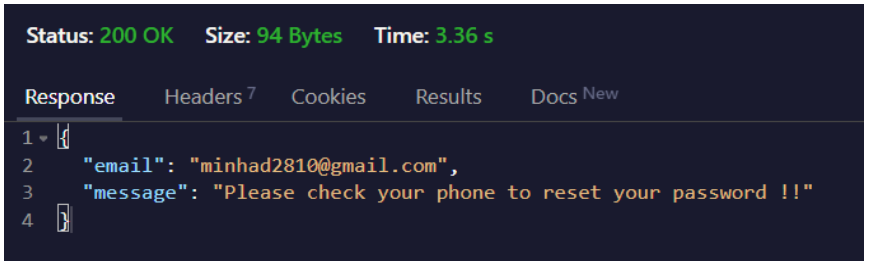


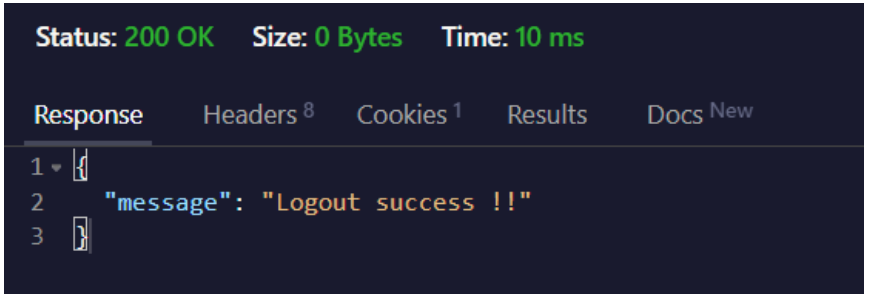

Figure 99: View booking detail of user.

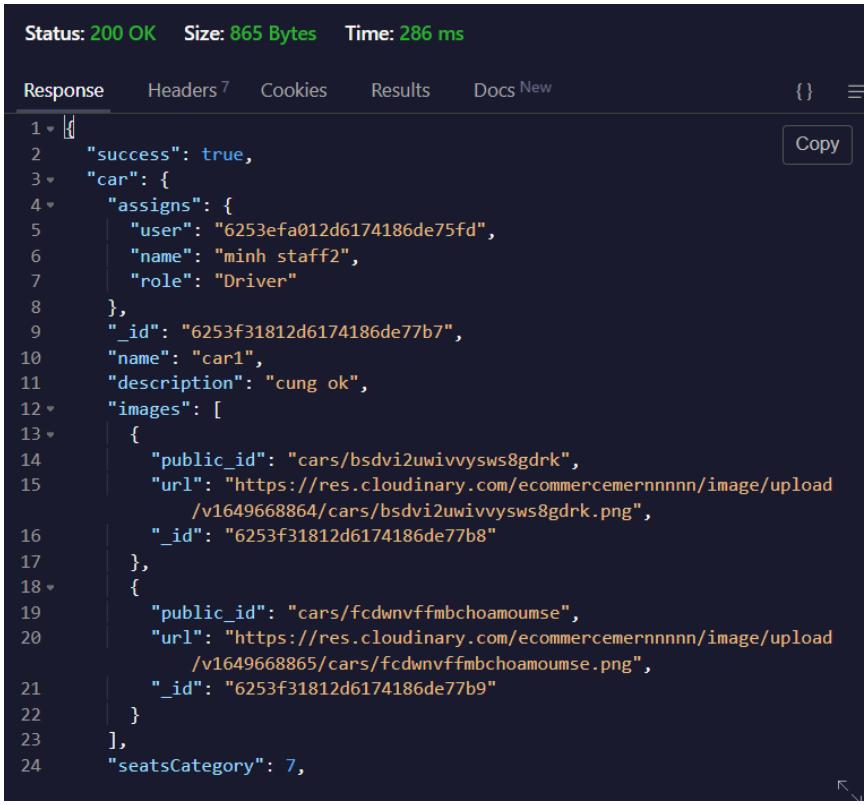
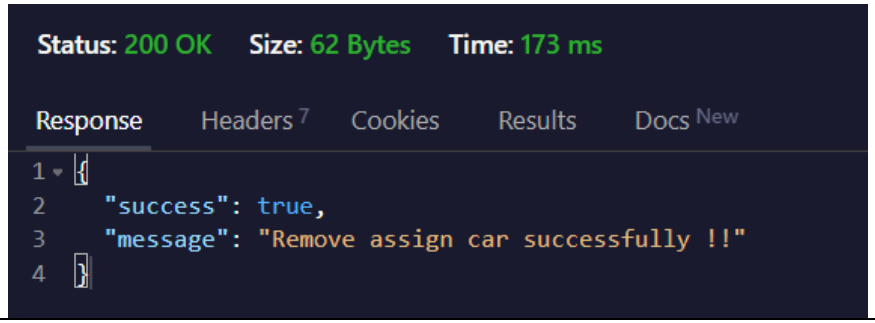
# 10. Testing:

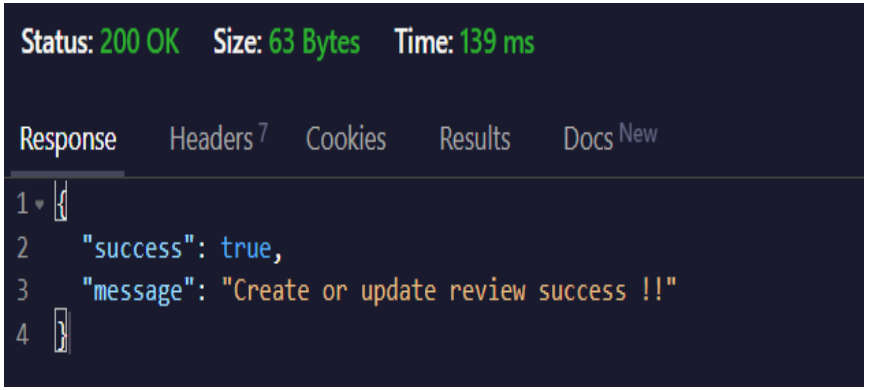
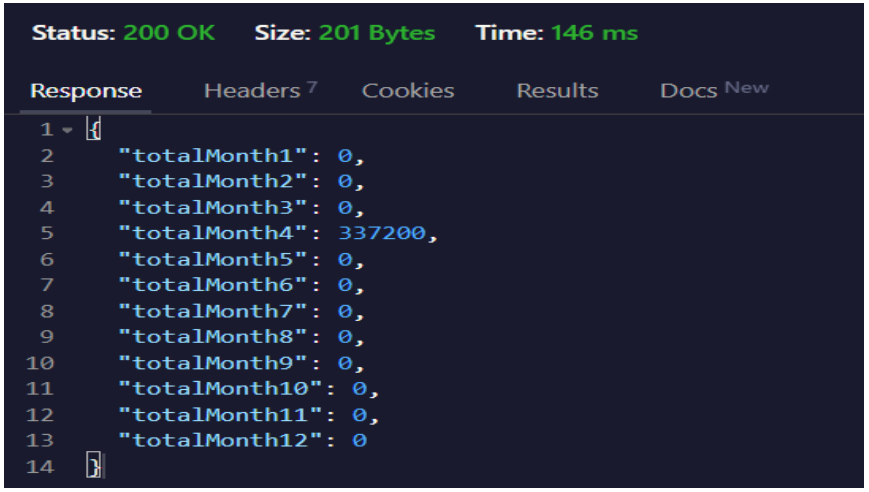
## 10.1. Test API:

Below I will test some the API of the TCAR application. And the functions that integrate between front-end and back-end will be tested in section 10.2.

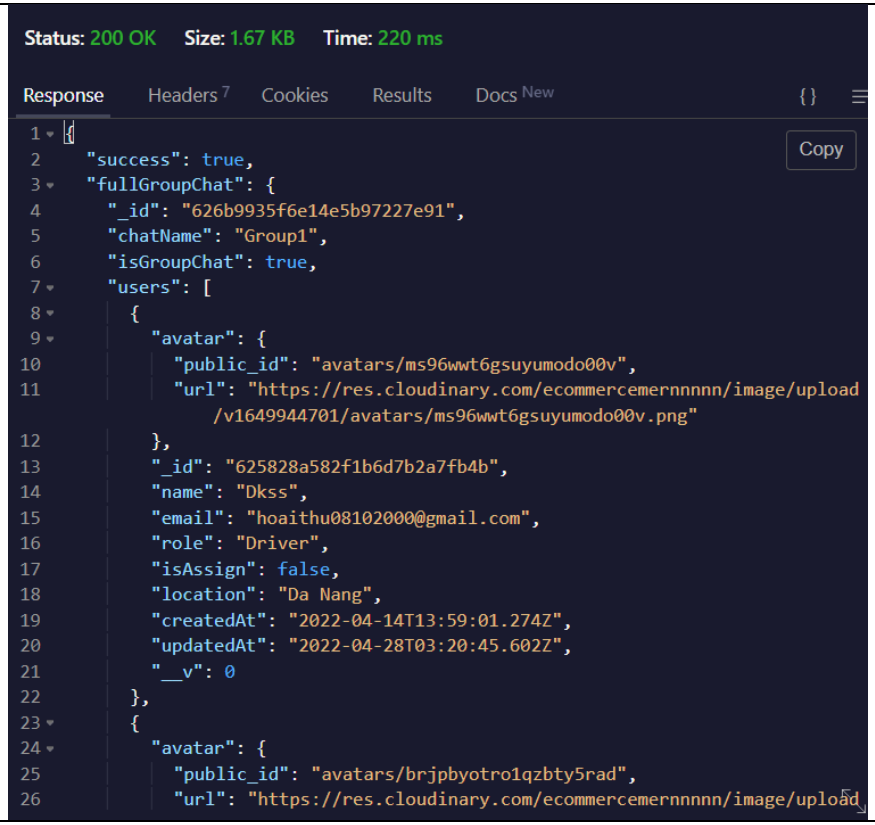
| ID | Test scenario | Link API | Input | Expected result | Evidence images | Test result | Status |
|---|---|---|---|---|---|---|---|
| 1 | User login | http://localhost:5000/api/login (Post) | { "email": "trandinhminh123@yahoo.com", "password": "123456" } | Returns status 200, message login success and information of user login. |   | User login success | Pass |
| 2 | User forgot password with OTP (phone) | http://localhost:5000/api/forgotPassword (Post) | { "email": "minhad2810@gmail.com", "method": "Phone", "phoneNumber": | Return status 200, returns the email of the user who forgot the password and the message asking the user to |   | User receive OTP (SMS) from phone | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | "+84905092786<br>"<br>} | check the phone. | | | |
| 3 | User logout | http://localhost:5000/api/logout<br>(Get) | N/A | Return message “Logout success !!” and status 200. |  | User logout success | Pass |
| 4 | Admin get all car | http://localhost:5000/api/cars/getAdAllCars<br>(Get) | N/A | Return car all car count, pagination number of car show, all cars and status 200. |  | Admin get all car success | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Staff assign driver to car | http://localhost:5000/api/cars/assign (Post) | { "carId": "6253f31812d6174186de77b7", "userId": "6253efa012d6174186de75fd" } | Return status true (200), and car assign with driver assign. |  | Assign car to driver success | Pass |
| 6 | Staff remove assign driver to car | http://localhost:5000/api/cars/removeAssign/6253f31812d6174186de77 | N/A | Return status true (200) and message” Remove assign car successfully !!” |  | Remove assign car for driver success | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | b7 (Delete) | | | | | |
| 7 | User review car | http://localhost:5000/api/cars/reviewCreate (Put) | { "carId": "6253f31812d6174186de77b7", "driver": "cung dccc upss", "comment": "but also the comment", "rating": 3 } | Return status true (200) and message" Create or update review success !!" |  | User create review success | Pass |
| 8 | Admin get all total money each month | http://localhost:5000/api/booking/statisticsTotalAmountBooking (Get) | N/A | Return status true (200) and data of total money each month. |  | Admin get all total money each month success | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | Staff get all booking | http://localhost:5000/api/booking/getAllBooking (Get) | N/A | Return status true (200), total all price, books count of user, and all information detail of booking. |  | Staff get all booking with Infor detail and all total price of booking | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | Create a new group chat | http://localhost:5000/api/chat/createGroupChat (Post) | {<br>"name": "Group1",<br>"users": "[\"62298c0dd4a2056147275b4d\",\"621cd059b6a2c5ed651565b0\",\"625828a582f1b6d7b2a7fb4b\"]"<br>} | Return success (200) and data of new group chat |  | Create a new group chat success | Pass |

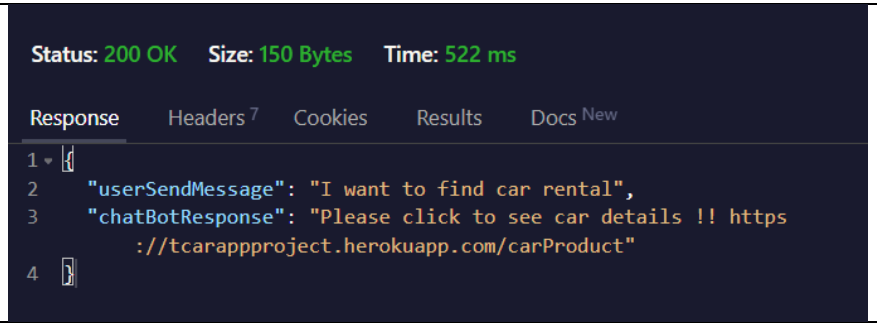
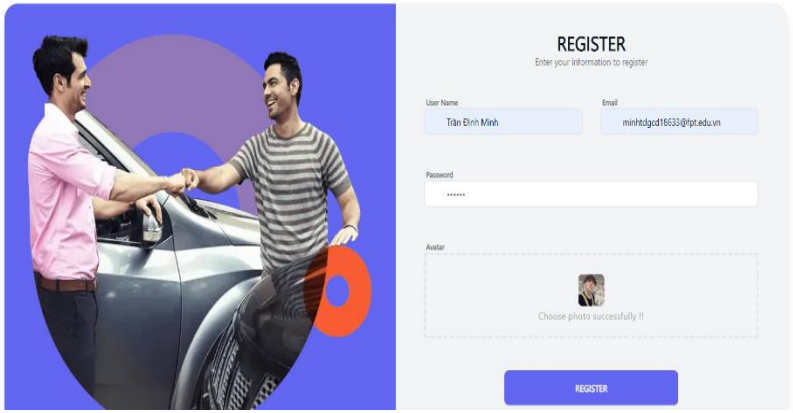
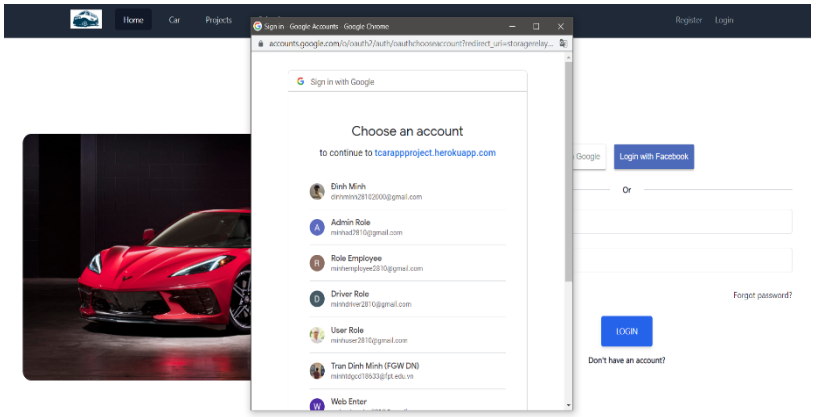
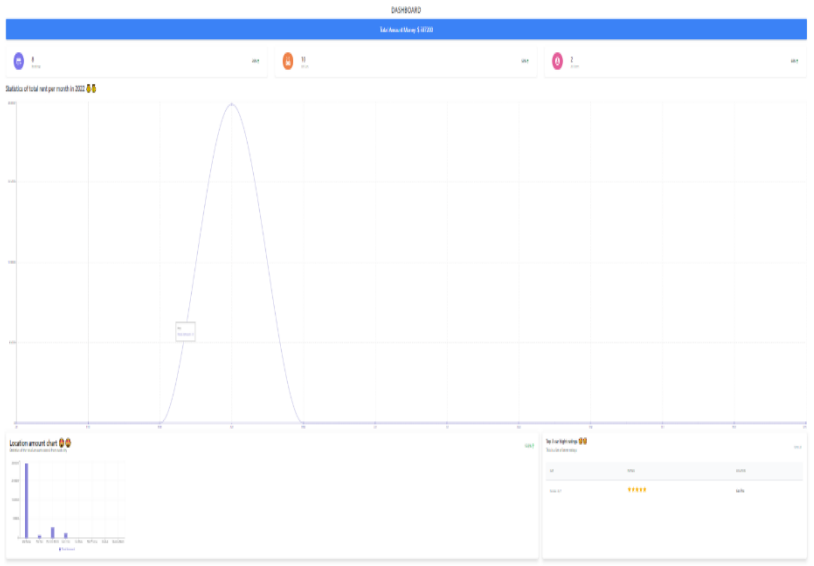
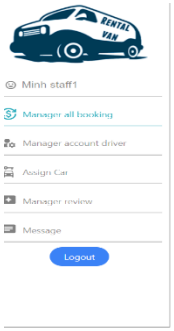
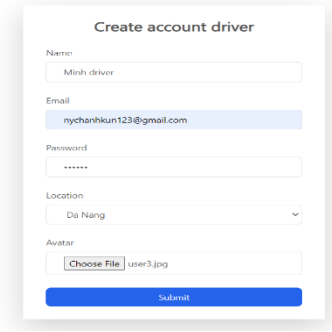
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | Send message | http://localhost:5000/api/message/sendMessage (Post) | { "content": "Hello", "chatId": "626b9935f6e14e5b97227e91" } | Return status (200), user sender data, content send, and group chat. |  | Send message to group chat success | Pass |
| 12 | User ask chatbot | http://localhost:5000/api/chatbot/sendMessageToChatBot (Post) | { "text": "I want to find car rental" } | Return status 200 and see user send message and chatbot response. |  | Chatbot reply ask of user success | Pass |

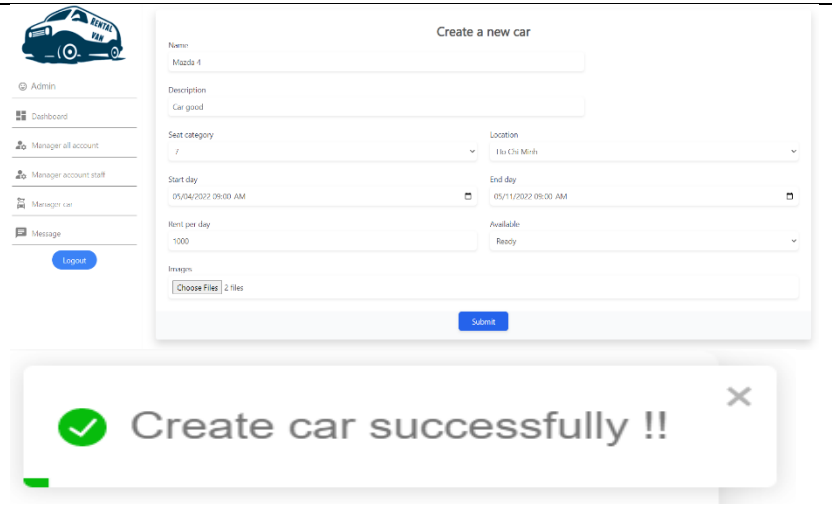
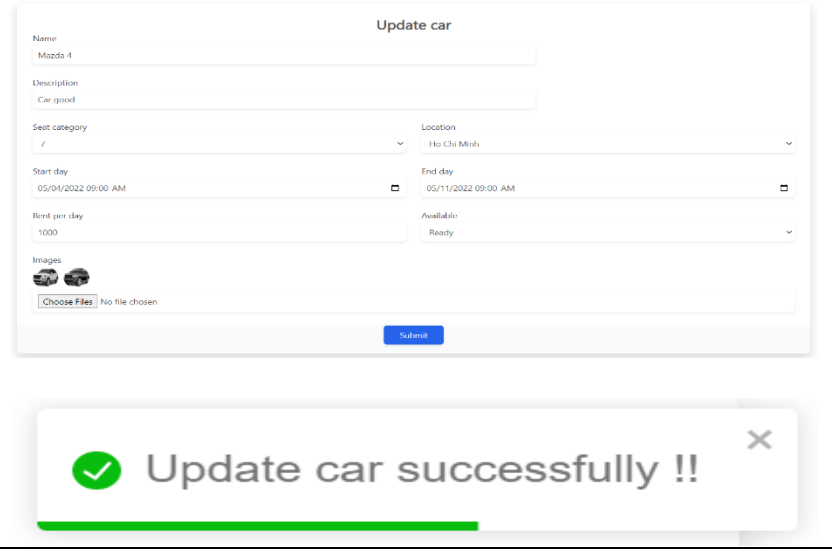
Table 6: Test API.

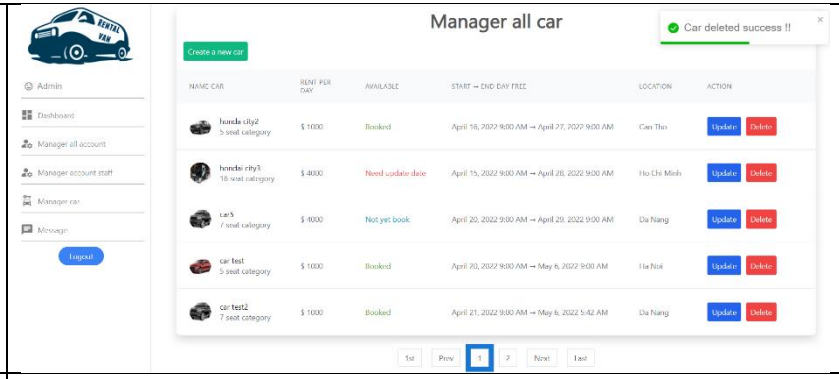
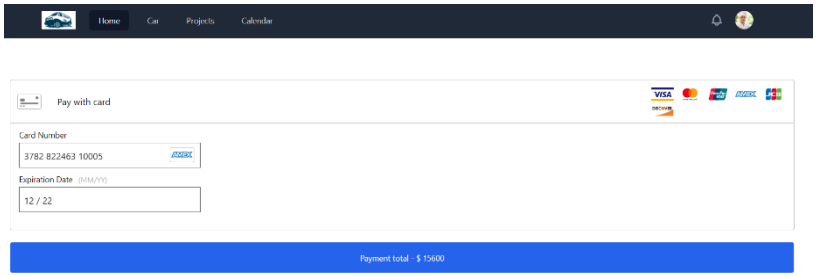
### 10.2. Integrated test:

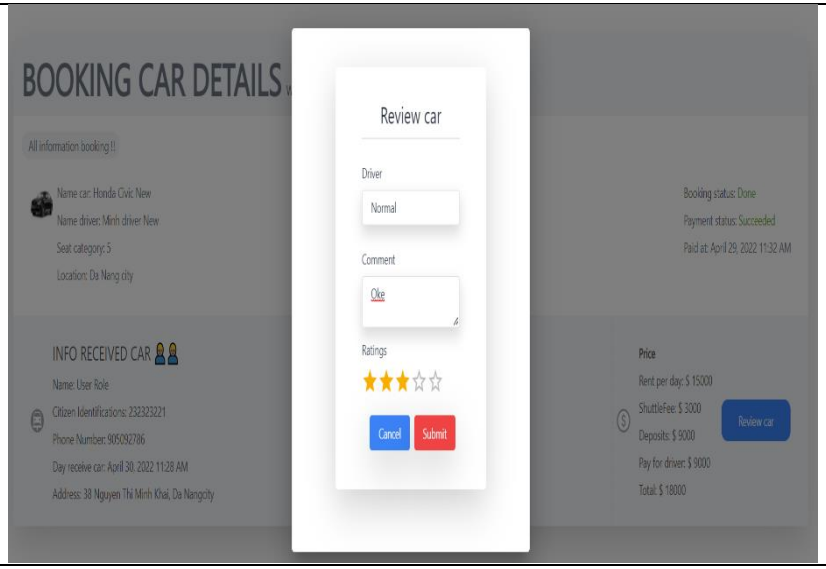
Below, I will test some integration functions between front-end and back-end:

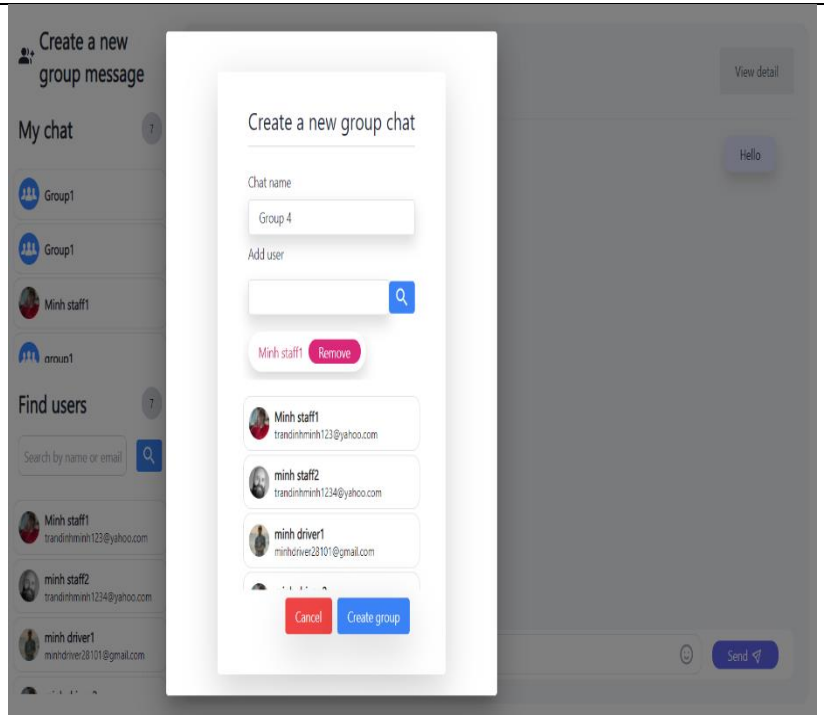
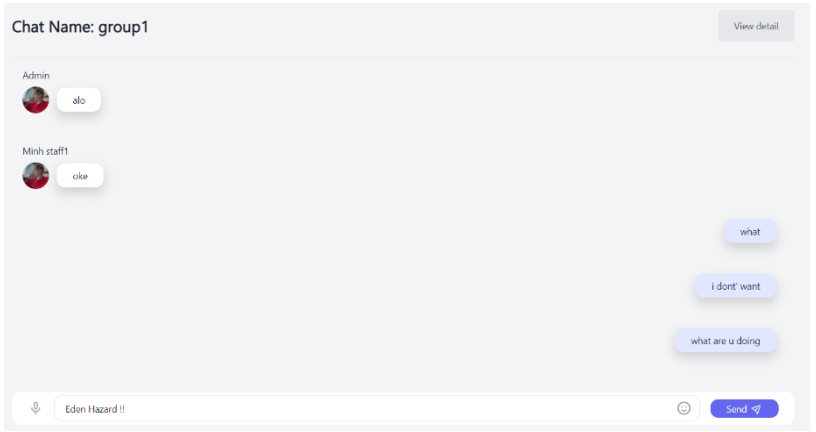
| ID | Test scenario | Steps | Input | Expected result | Evidence images | Test result | Status |
|---|---|---|---|---|---|---|---|
| 1 | Register | 1. Click the button register.<br>2. Enter the data in the input fields to form register.<br>3. Receive email.<br>4. Go email to active account. | Username: TranDinhMinh ,email: minhtdgcd18633@fpt.edu.vn, password:DinhMinh, choose image | Receive email, click link active account in email and navigate to register success page. |  | As expected result | Pass |
| 2 | Login with google | 1. Click the button login with google.<br>2. Display modals select account or login account and real email password.<br>3. Click login. | Emai: Trandinhminh28102000@gmail.com<br><br>Pass: DinhMinh281 | Login with google success and navigate to home page with role corresponding |  | Login with google success and navigate to home page with role corresponding | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | Admin see dashboard | 1. Login with account admin.<br>2. System auto navigate to home page of admin (dashboard) | N/A | Admin see total user account, booking and all cars. Admin can see the graph of the total amount at each month and total amount at each location and see car have ratings hight. |  | As expected result | Pass |
| 4 | Staff create account staff | 1. Click button manager account driver.<br>2. Navigate to all account driver. | Name: Minh driver, email: nychanhkun123@gmail.com, password:123456, Location: Da Nang, choose image avatar. | Create account driver success , and toast message “Create account driver success” for staff. |  <br>  | As expected result | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Admin create car | 1. Click button create car.<br>2. Enter the data in the input fields to form create car. | Name: Mazda4, Description: Car good, Seat category: 7, Location: Da Nang, choose start day and end day, rent per day: 1000, Available: Ready, choose images car. | Admin can create car success and toast message “Create car successfully !!” for admin. |  | As expected result | Pass |
| 6 | Admin update car | 1. Click button update car.<br>2. Enter the data in the input fields to form create car. | Just enter the fields that the admin wants to update | Update car success and toast message “Update car successfully !!” for admin. |  | As expected result | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | Admin delete car | 1. Click button manager all cars.<br>2. See all car.<br>3. Click button “Delete”. | N/A | Delete car and toast message” Car deleted success !!” |  | Delete car success and toast message” Car deleted success !!” | Pass |
| 8 | Pay with Braintree | 1. Choose car rental.<br>2. Confirm information rental car.<br>3. Payment. | CardNumber: 3782 822463 10005 Expiration Date: 12/22 | Payment success and navigate to my booking (user) page. |  | As expected result | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | Review car | 1. Status booking car must done.<br>2. User click button “My booking”.<br>3. Click detail booking.<br>4. Click button “Review car”<br>5. Enter data and click button “Submit”. | Driver: choose values in dropdown, Comment text, choose rating. | Review success with toast message and navigate to home page. |  | As expected result | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | Create a new group chat | 1. User click “Create a new group message”.<br>2. System wil show a modal to create group chat.<br>3. User enter name of chat name group.<br>4. User search users who user wants add to group chat.<br>5. Click button “Create group” to create group chat. | ChatName: Group1, choose user. | Create group chat and toast message for user |  | Create group chat and toast message for user success. | Pass |
| 11 | Send message | 1. Click to group chat.<br>2. Send message. | Enter a new message | Send message to chat success |  | Send message to chat success | Pass |

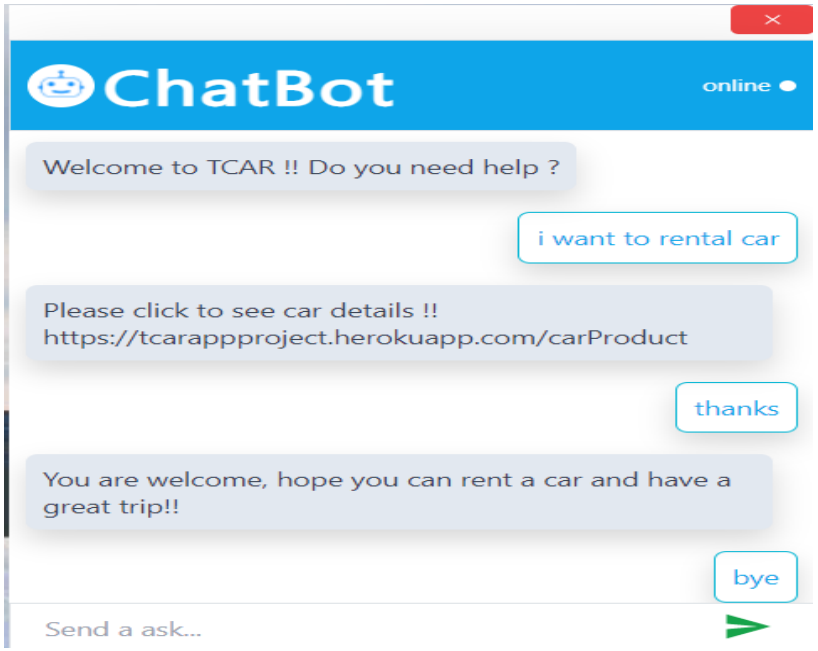
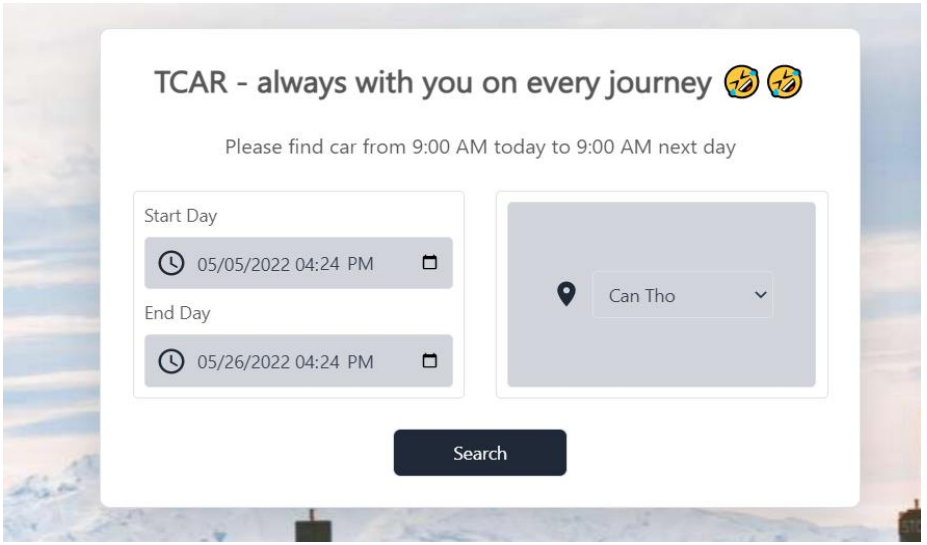
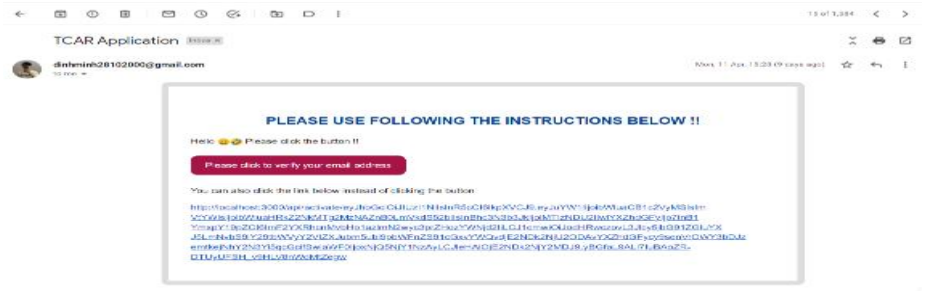
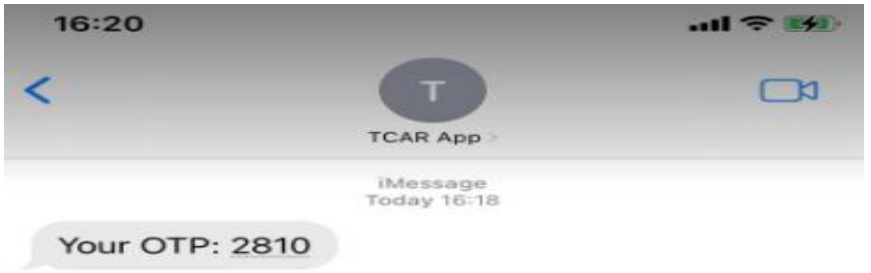
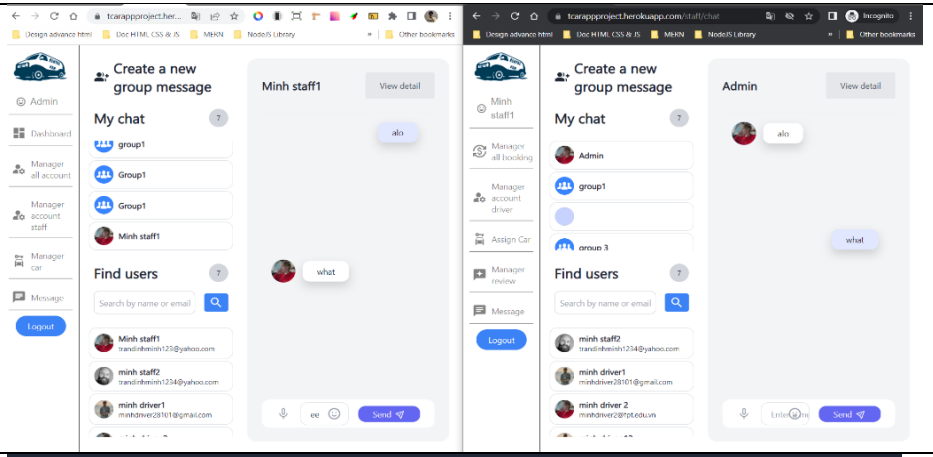
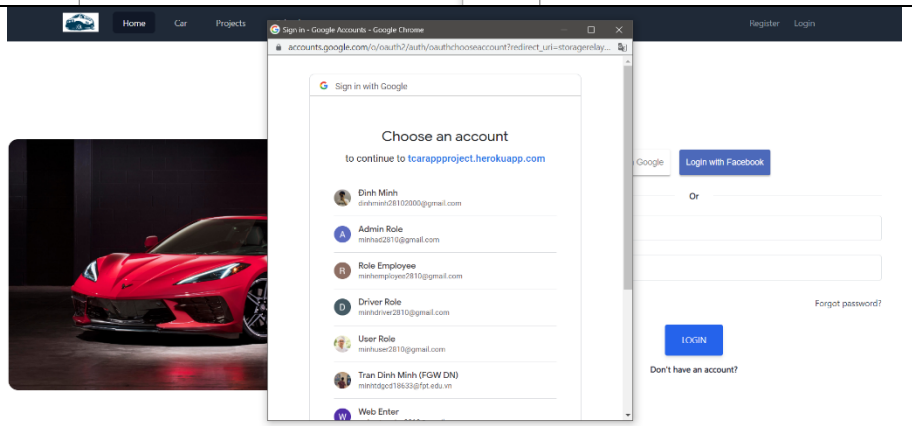
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 | Send ask to chatbot | 1. User go to home page.<br>2. Chatbot will show for user.<br>3. Chat bot will say “Welcome to TCAR !! Do you need help?”<br>3. User send a ask to chatbot. | Send any ask to chatbot | User send ask to chatbot success and chatbot reply ask of user. |  | As expected result | Pass |

Table 7: Integrated test.

### 10.3. System test:

After successful deployment, I will test some functions to run well? how is the runtime and do 3rd parties have good app support?

| ID | Test scenario | Steps | Input | Expected result | Evidence images | Test result | Status |
|---|---|---|---|---|---|---|---|
| 1 | User find car | 1. User go to home page.<br>2. User select date and location to choose car | Choose stat day, end day, and location | Go to all cars page with the time the user is looking for. Time to find car 0.25s. |  | Go to all cars page with the time the user is looking for. Time to find car 0.25s. | Pass |
| 2 | Receive email to active account | 1. Go to register page.<br>2. Enter all fields of account registration data.<br>3. Go to email. | N/A | Send emails to users within 20 seconds |  | As expected result | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | Receive OTP from SMS | 1. User go to forgot password.<br>2. SMS send to phone's user. | N/A | Send SMS to phone users within 20 seconds |  | As expected result | Pass |
| 4 | Chat message real time | 1. Admin send message to staff.<br>2. Staff receive message from admin. | Admin or staff send a message | Send message success and real time message receiving time is 1s. |  | Send message success and real time message receiving time is 1s. No problems after deploying the app. | Pass |
| 5 | Login with google or facebook | 1. Click button login "Google" or "Facebook"<br>2. Enter data in modal google or facebook | Email: Minh@ gmail.com<br>Password: 123456 | 3rd parties (Facebook and Google) that support Heroku to login once deployed |  | Login success | Pass |

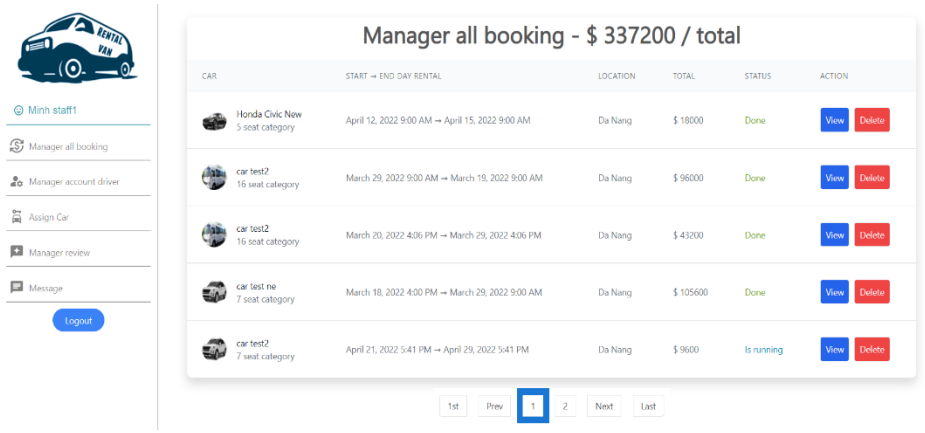
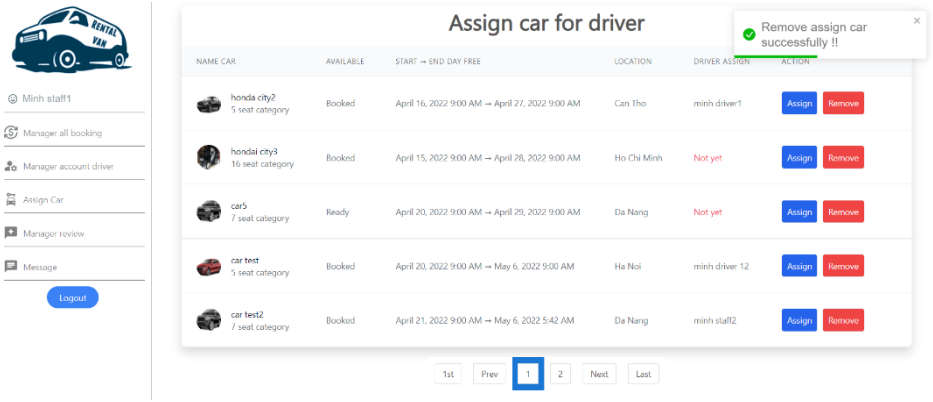
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | Staff manage all booking | 1. Login with account staff.<br>2. Click button “Manage all booking” | N/A | Staff see all booking, and status booking still automatically generates the booking status of the car based on the start and end time of the car rental user. |  | As expected result | Pass |
| 7 | Remove assign car to driver | 1. Click button “Assign Car”<br>2. View all car assign or not assign.<br>3. Click button “Remove” | N/A | Show all information of car assign or not assign, can remove assign car for driver. |  | Show all information of car assign or not assign, remove assign car for driver success | Pass |

Table 8: System test.

# 11. Evaluation:

## 11.1. Summarized key findings from the project:

### 11.1.1. Technical:

| Discovery | Justification |
|---|---|
| **Full-Stack (MERN)** | I researched quite a few full-stack developer toolkits, and I chose the MERN stack. This is because I have received advice from my predecessors that the toolkit is quick to learn and easy to develop. I learned how front-end can call api from server side. Using MongoDB will interact with JavaScript language better because MongoDB stores the data type as an object and JavaScript always works well with object and array problems, which helps me solve many problems when server interact with the database more. |
| **Front-end** | I have discovered a lot of good things when using ReactJS in combination with Redux to manage states easily. ReactJS syntax is based on pure html, so writing code becomes easy to understand and transparent. Calling the api just needs to use axios instead of before I used fetch() to call the api. Moreover, reactJS is also supported by many libraries that can fully serve the development of TCAR applications. It is also strongly supported by built-in design componets such as bootrap, antdesign, taiwindCSS and this helps me explore more about design forms to see which looks good and which one is not good yet. |
| **Back-end + MongoDB** | Before developing this application, I only understand about MVC model and this is my problem? But after completing the project, I understood what api is? how important it is and what it can do when developing a client- |

| | |
|---|---|
| | server model. The two-way data exchange between the client and the server needs to be confirmed and authorized by the server to protect the database data. And using MongoDB is a perfect combination with NodeJS using Javasript language, this makes the interactions when querying data easier when Javascirpt works strongly with arrays and objects to help interacting easy with collections of MongoDB. |
| **Deployment (Heroku)** | I have discovered that heroku strongly supports the MERN toolkit when deploying apps. Just connecting to GitHub and setting up a few commands I was able to deploy the TCAR application. Although, I only use the free service plan, but Heroku also supports app with stability when using the application, so I think Heroku is supporting the service for those who are just starting to learn about deploy is quite attentive and good. |

Table 9: Summarized key findings about technical.

### 11.1.2. Business:

Here's what I discovered about the business side of the project:

- Vietnam is considered an attractive tourist destination by famous landmarks such as Ba Na hill, Sword Lake, Ha Long Bay, and so on.
- Human demand for tourism is increasing. This leads to overloaded tourist car rental and this is a problem that needs to be solved so that passengers can use better travel services in Vietnam.
- The accident rate in Vietnam is quite high, so TCAR prioritizes and trains drivers to limit the worst risks to travel services. In particular, drivers at TCAR always put safety first with the motto "Safety is a companion on every trip".
- Furthermore, I have researched the problems of current car rental applications, from which to evaluate and to fix those problems so that TCAR becomes more complete before the product is released released to users.

## 11.2. Recommendations for future development:

| Future development | Justification |
|---|---|
| Map integration | In the future, I need to integrate maps for TCAR app. The purpose of the map integration is to help passengers find the car in more detail and clarity, the map will show which cars are available free time at the location that the user wants to rent. Furthermore, the integration of the map will help the authorized roles in the system to see the cars that are carrying passengers during the trip, whether the driver is carrying the passengers where they want, or if the driver is profitable Using passengers who are visiting at a certain place to takes the car to do his own work. |
| Use AI (Artificial Intelligence) | In the future, I think the system needs to assign an AI device to the driver's car with the purpose of immediately notifying the authorized departments in the TCAR system and the passengers' relatives when the car is in an accident. The AI device will link with the TCAR application, and it is possible that the Python language will be selected to develop the connection and integration of the AI device with the TCAR application. |
| Mobile app | For the tourist car rental system, I think I will develop a TCAR mobile app. This makes it possible for users to use the application on more platforms instead of just using it on a web browser. And if I have developed the TCAR app on the phone platform, I will develop more login functions with qr code and payment that link with VietinBank iPay or Techcombank ipay app. |
| Improve UX & UI | In the future, TCAR will probably be improved on UX &UI to improve the quality of the application. Rearrange the layout with |

| | |
|---|---|
| | the right colors to create a unique highlight for the application. Animations will be used to make the interface more polished and beautiful. |
| Performance | When the application is used by too many users, it will bring the successful values of TCAR. As a result, the system will need to improve the performance of the application to avoid problems such as slow response times. With the development of DevOps that means APM can be used throughout the product development lifecycle to increase the frequency of releases while also improving the quality of those releases. |

Table 10: Recommendations for future development.

## 11.3. Project evaluation:

After completing the project, the TCAR application has completed all the functions that the goal set out. With not much time to develop the project, but I still fully meet the requirements of the project so that the project can achieve revenue and meet the demand for car rental services in Vietnam when Vietnam is back stronger about the tourism industry after difficult times by covid-19. From the problems of existing applications, the project did not follow the dock but developed a brand-new system to provide the best user experience. With the introduction of TCAR, solving car rental service problems became easy because of my professional investment in this project.

Car search becomes easier when the system integrates the chatbot into the application. Chatbot will support all user queries when users have problems. Those with permissions in the system can communicate with each other right on the application instead of having to use other applications such as Facebook, Zalo, Viber, and so on. This helps those with rights in the system have an equally good experience when using the TCAR application instead of just satisfying the needs of customers without caring about TCAR's employees. In addition, the customer's bank credit card is also anonymized to those with authority in the system, and the third party involved will assist in the security of the user's credit card, which helps the user trust TCAR services and avoid the corruption that TCAR employees can do for their own gain.

Besides the positive things that the system has developed, the time management and making plans for the project are limited. Although, I have given a specific time for the project plans, but when developing the project, there are some unexpected problems, that is, I have covid-19 or

have a code error, which leads to the failure of the project is delayed in project development. In addition, I need to analyze plans and requirements specification in more detail to avoid wasting time in product development. Next, the application has not yet been able to integrate the map so that users can find the car more easily and the rights holders in the system can manage the running bookings better. The user interface is normal and needs to be improved. Finally, I should develop a TCAR mobile application.

In general, the project has followed the right trajectory with the initial goals and the application can completely compete fairly with current car rental applications.

## 11.4. Personal evaluation:

After completing the project, I made a MERN application, which was unexpected because I have never developed an application from front-end to back-end and even deploy. In terms of positivity, I have tried my best to research and study to defeat all doubts about myself. This helps me to try and develop more in the future when doing real software projects. My ability to self-study and find documentation is also enhanced when encountering bug code.

The bad side when developing this project is the time management. The time management of the initial targets for each requirement was delayed a lot because I still could not cover the requirements specification of the system which led to many problems. With my limited knowledge, I have not been able to develop a more outstanding function such as integrating maps or assigning traffic devices when the vehicle is in a traffic accident.

Finally, after this project, I also learned a lot of core things about developing a website with the MERN Stack toolkit. I learned perseverance and effort to overcome difficulties during the development of this project. Overall, I have tried to learn and recognize my weaknesses so that I can improve on them in the future. I am proud of my efforts and making the impossible possible. Furthermore, I also learned how to properly research references successfully and this was the key for me to achieve the purpose of the project with a complete application.

## 11.5. Conclusion:

After completing this project, I have gained a lot of knowledge and experience when developing a software application. This project required me to do everything from collecting user information, specifying requirements that the system requires, clearly defining the project development time. Next, I must research which technology is right for the project to develop. During development, I must learn and fix the problems that I face while developing the project. Finally, I would like to thank Mr. Tran Trong Minh and Mr. Hoang Nhu Vinh for supporting and answering all my questions to complete this project. And this is really the end, I would like to grateful and thank the University of Greenwich for giving me a great learning environment for me to take the next step and pursue my IT career in the future.

## 12. References:


Adetokunbo A.A. Adenowo, B. A. A., 2013. Software Engineering Methodologies. *A Review of the Waterfall Model and Object-Oriented Approach,* 4(7), pp. 429-430.

Braintree, 2022. *Frequently Asked Questions.* [Online]
Available at: https://www.braintreepayments.com/faq
[Accessed 22 4 2022].

Cahill, A., Fall 2018. Sustainable Tourism Practices in Vietnam. *The Influence of Institutions and Case Study of Sapa's Growing Tourism Industry,* p. 7.

Cloud, G., 2022. *Dialogflow.* [Online]
Available at:
https://cloud.google.com/dialogflow/docs#:~:text=Dialogflow%20is%20a%20natural%20language,response%20system%2C%20and%20so%20on.
[Accessed 22 4 2022].

current, T., 2022. *What is Twilio? An introduction to the leading customer engagement platform.* [Online]
Available at: https://www.twilio.com/the-current/what-is-twilio-how-does-it-work
[Accessed 22 4 2022].

FreshBooks, 2021. *What Is Stripe and How Does it Work?.* [Online]
Available at: https://www.freshbooks.com/hub/payments/what-is-stripe
[Accessed 22 4 2022].

Holdings, E., 2018. *Enterprise Rent-A-Car Opens for Business in Vietnam.* [Online]
Available at: https://www.prnewswire.com/news-releases/enterprise-rent-a-car-opens-for-business-in-vietnam-300757256.html
[Accessed 15 4 2022].

Kambil, A., 2008. What is your Web 5.0 strategy?. *An edited version of this viewpoint appeared in the Journal of Business Strategy 2008,* 29(6), pp. 4-5.

Ken, 2017. Vietnam Car Rental Market Outlook to 2021. *Rising Tourism and Trend of Mobile Booking for Rental Cars to Drive Market,* 2(1), p. 5.

Ken, 2017. Vietnam Car Rental Market Outlook to 2021. *Rising Tourism and Trend of Mobile Booking for Rental Cars to Drive Market,* 2(1), p. 5.

Ken, 2019. *Growing Mobile Applications coupled with Rise in Number of Tourists in Vietnam Car Rental Market.* [Online]
Available at: https://wtocenter.vn/chuyen-de/13872-growing-car-rental-companies-mobile-

applications-coupled-with-rise-in-number-of-tourists-will-drive-the-growth-of-vietnam-car-rental-market-ken-research
[Accessed 15 4 2022].

Ken, 2019. *Growing Mobile Applications coupled with Rise in Number of Tourists in Vietnam Car Rental Market.* [Online]
Available at: https://wtocenter.vn/chuyen-de/13872-growing-car-rental-companies-mobile-applications-coupled-with-rise-in-number-of-tourists-will-drive-the-growth-of-vietnam-car-rental-market-ken-research
[Accessed 27 10 2021].

Labs, O., 2022. *What Is the Express Node.js Framework?.* [Online]
Available at: https://heynode.com/tutorial/what-express-nodejs-framework/
[Accessed 15 4 2022].

Margaux Constantin, M. F. a. T. L., 2021. Đổi mới ngành du lịch. *Việt Nam có thể đẩy nhanh tốc độ phục hồi như thế nào,* pp. 2-4.

MDN, 2022. *What is JavaScript?.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript?retiredLocale=vi
[Accessed 15 4 2022].

Nam, B. ả. V., 2022. *Du lịch Việt Nam tự tin mở cửa trở lại, hướng đến mục tiêu đón 5 triệu khách quốc tế trong năm 2022.* [Online]
Available at: https://vietnam.vnanet.vn/vietnamese/tin-van/du-lich-viet-nam-tu-tin-mo-cua-tro-lai-huong-den-muc-tieu-don-5-trieu-khach-quoc-te-trong-nam-2022-289190.html
[Accessed 15 4 2022].

News, G., 2022. *Các địa phương cần 'tăng tốc' tiêm mũi 3 vaccine phòng COVID-19.* [Online]
Available at: https://baochinhphu.vn/cac-dia-phuong-can-tang-toc-tiem-mui-3-vaccine-phong-covid-19-102220326175552384.htm
[Accessed 15 4 2022].

Nguyen Thi Hoa, D. T. N. H., 2021. Vietnam Tourism Services Development During and after Covid 19 Pandemic. *Situation and Solutions,* 11(No 3 (2021)), pp. 26-27.

Nguyen Thi Hoa, D. T. N. H., 2021. Vietnam Tourism Services Development During and after Covid 19 Pandemic. *Situation and Solutions,* 11(No.3 (2021)), pp. 26-27.

Nodemailer, 2022. *NODEMAILER.* [Online]
Available at: https://nodemailer.com/about/
[Accessed 22 4 2022].

Peerbit, 2022. *The benefits of ReactJS and reasons to choose it for your project.* [Online]
Available at: https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html
[Accessed 4 15 2022].

Rentalcars.com, 2022. *About us.* [Online]
Available at: https://www.rentalcars.com/en/about/
[Accessed 19 4 2022].

Research, K., 2018. Car Rentals (Self Drive) Market in Vietnam to 2022. *Fleet Size, Rental Occasion and Days, Utilization Rate and Average Revenue Analytics,* p. 37.

research, M., 2021. Car Rental (Destination) in Vietnam. pp. 10-15.

Staticta, 2022. *Most used web frameworks among developers worldwide, as of 2021.* [Online]
Available at: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/#:~:text=React.,reported%20to%20be%20using%20React.
[Accessed 15 4 2022].

TechTarget, 2022. *Google Cloud.* [Online]
Available at: https://www.techtarget.com/searchcloudcomputing/definition/Google-Cloud-Platform
[Accessed 22 4 2022].

Thu, P., 2018. *Car rental market accelerates.* [Online]
Available at: https://propertycloud.asia/news/1684
[Accessed 15 4 2022].

Turo, 2022. *About Turo.* [Online]
Available at: https://turo.com/us/en/about
[Accessed 19 4 2022].

TutorialSpoint, 2022. *Node.js - Introduction.* [Online]
Available at: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm#:~:text=is%20as%20follows%20%E2%88%92-,Node.,Node.
[Accessed 15 4 2022].

Wikipedia, 2022. *Đại dịch COVID-19 tại Việt Nam.* [Online]
Available at: https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1i_d%E1%BB%8Bch_COVID-19_t%E1%BA%A1i_Vi%E1%BB%87t_Nam#:~:text=V%C3%A0o%20ng%C3%A0y%2023%20th%C3%A1ng%201,ch%E1%BA%BF%20t%E1%BB%B1%20do%20di%20chuy%E1%BB%83n.
[Accessed 15 4 2022].

Wikipedia, 2022. *MongoDB.* [Online]
Available at: https://en.wikipedia.org/wiki/MongoDB
[Accessed 15 4 2022].

# 13. Appendix A – Project Proposal:

## 1. Overview:

In Vietnam, tourism is considered a spearhead economic sector with rich potential and future development. Thanks to the places and rich elements, the places attract a lot of domestic and international tourists. According to Nguyen Thi Hoa, from 2016-2019, the number of international visitors to Vietnam is increasing day by day with a huge number from 72 million to 103 million visitors with a high growth rate (Nguyen Thi Hoa, 2021). These are considered factors that show the hotness of the tourism industry in Vietnam.

Currently, the demand for car rental is increasing, leading to the car rental service in Vietnam sometimes overloaded. The number of people using car rental services has skyrocketed year by year with 34% and 48% annual growth based on (Ken, 2019). According to Ken, there are still many tourists who have not caught up with the trend of switching to online car booking through applications although the car rental service exists and is owned by MGM, SaiGon Star, and so on companies, but it still cannot meet the needs of tourists (Ken, 2017). In order to make it possible for all travelers to book car rental services and pay online, I want to develop an application with ease of use and professionalism on the top so that users can easily easy use to the application. In particular, unexpected accident situations are always a question raised and put on top to be solved when passengers join the car rental service.

With the above factors in mind, I decided to develop a tourist car rental application with the aim of supporting the country's tourism industry to develop further and find a solution to overcome the unexpected accident situation for customers with more than 300 tourists agree me to develop this app. The app is called Tourist Car Rental App (TCAR).

## 2. Aim:

This project is developed with the aim of developing the tourist car rental service to become more professional, to bring friendliness, minimize the problem of overloading when renting a tourist car and to ensure the safety of each passenger.

## 3. Objectives:

### 3.1. Literature review:

#### 3.1.1. Research identifies factors:

I need to study and define software development techniques. Identify user needs for application development. Use literature review to determine the exact scope of the system. The main purpose of doing this app development was to give me a better understanding of how the front-end and back-end of an application work. In addition, understanding the standards and application scopes will help me develop the system according to user requirements. Moreover, survey the needs of customers who want the application to develop through data from annual tourists in Vietnam. Investigate how successful the system is with its initial goals such as reducing accidents and overloading with car rental services.

#### 3.2.2. Research technology:

I will conduct research on web application technologies with the goal of cross-platform development. Clearly define the system's decentralization along with security. Use the included API services in tandem with NodeJS. Finally, I also researched the trend of easy online payment with the goal of safety for each traveler when using the application's services.

#### 3.1.3. Collect and analyze requirements:

Collect the necessary information from national tourists or even foreign tourists. I will proceed to create a google form questionnaire to cover all the opinions from the survey participants. From these factors, I will conduct requirements analysis, then draw the left and right side for the application.

### 3.2. Development:

#### 3.2.1. Design for app:

- Use Case Diagram: Used to describe user interaction with application features. It makes it easy for me to identify the functions that the application needs to interact the user requirements.

- Entity Relationship Diagram: Use ERD for the purpose of representing the entities in the database, and the relationship between them. The elements that link the tables together help create the logic to connect the data to the application.

- Data Flow Diagram: DFD helps to define the information flows of the application. The information to be changed and where the information is stored is clearly defined to avoid application development risks. Design step-by-step according to the initial analysis requirements.

- Actor map: I use actor map to clearly define the links between organizations and individuals for TCAR. Pursuant to section 3.1.3, clear data from reviews and studies as well as implicit knowledge from travelers has been collected so it will assist me with many agents, as well as many supporting data to drive the development of the application.

- Design and prepare content for app: I will use wireframe to develop the front-end for the project. The project will be beautifully designed with clear, eye-catching split layouts intended to not confuse users with front-end layouts. The content will be professionally written with clear and minimal text and lots of images or videos so that users feel satisfied with the front-end of the application.

**3.2.2. Choose technical for app:**

The application will be developed with the latest technologies to provide the best user experience. On the front-end side, I will use HTML, CSS, JS, ReactJS. On the back-end side, I would use NodeJS. MongoDB will be the project's database.

**3.2.3. Built interface and server for app:**

Conduct reference to UI & UX websites with beautiful layouts to cook for the best project experience. Then proceed to develop necessary functions such as registration, login, information search, payment, call a loved one if an accident happens, admin can manage lower roles in the system, decentralization, and so on.

**3.3. Testing and deployment:**

**3.3.1. Testing:**

After the application development process, I will conduct testing to see if the system is working according to the original goal. If the system encounters errors or problems arise, I will fix errors and improve the product as soon as possible to keep up with the project schedule. A final stage of testing will perform is acceptance testing, with the participation of the user in the main role to determine whether the software system meets the user's requirements or not.

**3.3.2. Deployment:**

I will deploy the application with Heroku, Glitch, Google Cloud Platform, AWS, or Netlify based on the right platform for my system.

**3.4. Evaluation:**

Evaluate whether the functions have been completed against the original goal? Is the application functioning properly when interacting with the user? Identify the risks when the application is officially released and look for bugs that may arise. If system failures are identified, corrective

action should be taken immediately. If the product has been completed and there are no problems, then evaluate what elements the application is missing, need maintenance and what needs to be improved so that the application can develop more in the future.

## 4. Legal, social, ethical, and professional:

### 4.1. Legal:

The purpose of developing TCAR is to avoid overcrowding of passenger cars, make payment easier and avoid the risk of accidents leading to loss of life for tourists. Basically, the project will not face many legal difficulties. The authorities in the tourism and software sectors have agreed to develop the application based on the contractual agreement between the parties. The project will be implemented in the forms determined by the competent authority. Software developers and people involved in the tourism industry join and analyze to develop the application because this is a project based on the foundation of the growing tourism industry in Vietnam. I also promise we do not use the data for malicious purposes.

### 4.2. Social:

TCAR application helps support the country's tourism industry to develop further and find solutions to overcome unexpected accident situations for customers. Bring satisfaction to both domestic and foreign customers. Paying money also becomes easy through banking services and security to avoid unnecessary risks. The application will be used in all cities of Vietnam.

### 4.3. Ethical and professional:

Ethical issues will take precedence, I will agree to a contract to avoid issues such as harmful actions, patents, copyright, trade secrets, liability, piracy. If the application encounters any problem, the user can completely complain and refuse to use the application for any reason. Moreover, TCAR will also meet the information security needs for each user's account and authorized officers in the system.

## 5. Planning:

Here is table about the tasks to be done:

| No | Task | Description | Step Time | Process time |
|---|---|---|---|---|
| 1 | Research | User data collection and receive | 7 days | 30 days |
| | | Research UI & UX | 7 days | |

| | | | | |
|---|---|---|---|---|
| | | Research functional requirements | 7 days | |
| | | Research technical stack | 7 days | |
| 2 | Application development | Analyze user requirements | 7 days | 111 days |
| | | Identify and overcome risks | 7 days | |
| | | Design UI & UX, ERD, DFD, and actor map | 16 days | |
| | | Use the right technical stack and develop the application | 60 days | |
| | | Testing and fix problems | 14 days | |
| | | Deployment and maintenance | 7 days | |
| 3 | Evaluation | Identify the achievements set out initially after completing the application | 5 days | 14 days |
| | | Assess risks and solves during application development | 4 days | |
| | | The benefits that the system can bring and the limitations of app | 2 days | |
| | | Evaluate what elements the application lacks and what needs to be improved so that the application can develop more in the future | 3 days | |

Table 11: Task project.

Here is the progress of the project development plan:

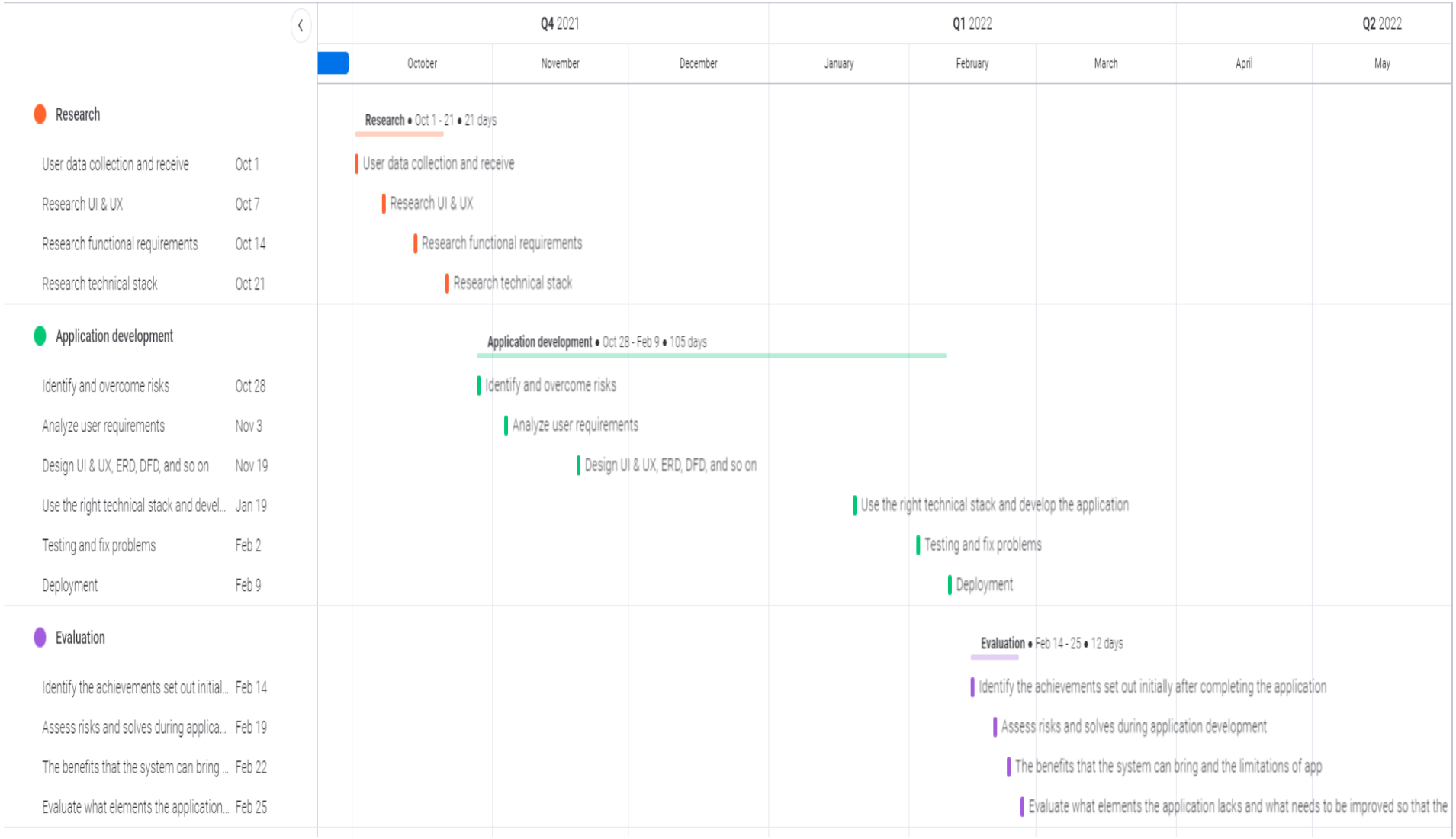



Figure 100: Gantt chart.

## 14. Appendix B - Sitemap:

This is sitemap of TCAR:

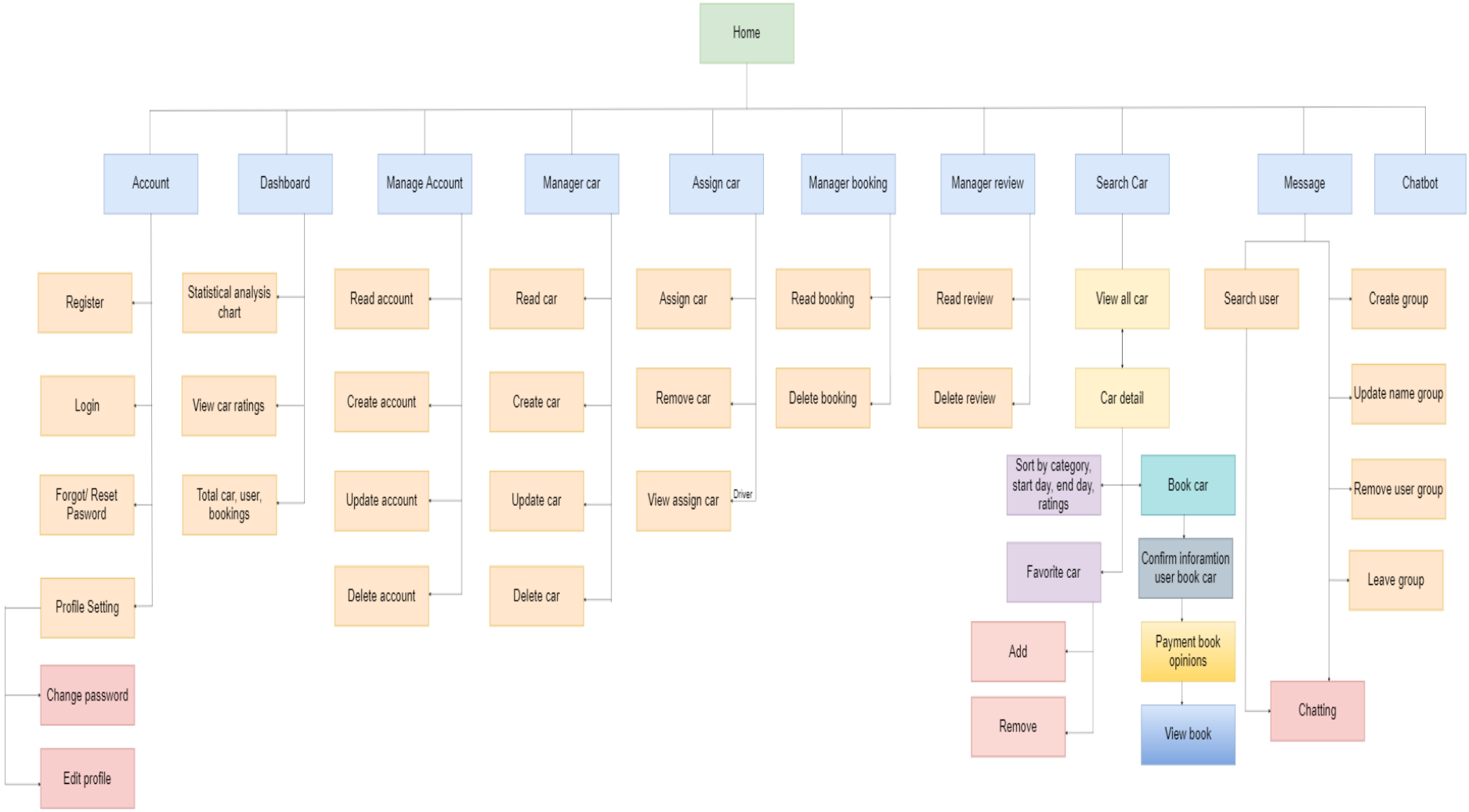



Figure 101: Sitemap of TCAR.