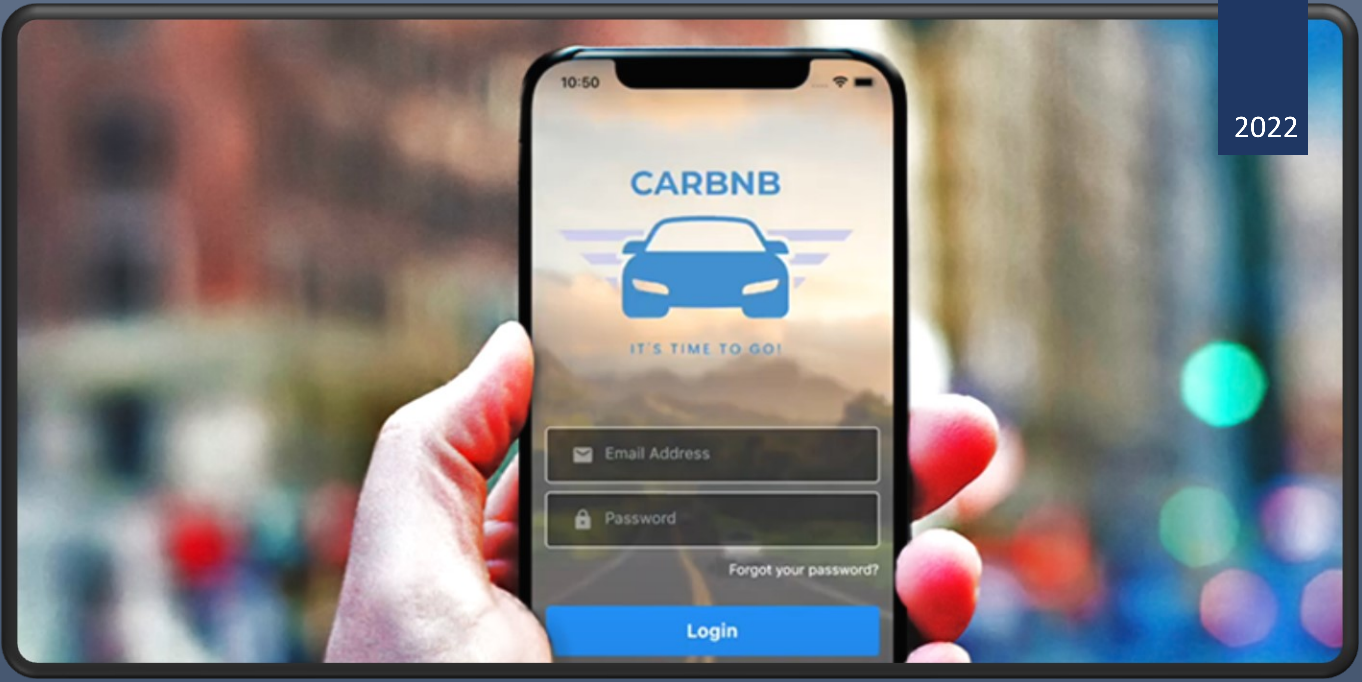
2022



CarBnB

Final Documentation

Building Web and Mobile Apps
Prof. Johannes Konert
Master seminar WS 2021/22
Hochschule Fulda

Prepared by Group 2A (CodeBusters)

Joshua Rahimi	**Ahmed Shahid**	**Obada Altinawi**	**Julian Günther**	**Julia Nürnberg**

-CONTENTS-

Abstract

CarBNB is a private car sharing platform inspired by the online accommodation marketplace AirBNB. The principle of the app is to rent out private cars to potential customers. The customer can use a filter function (e.g. by location, model or price) to find the car he/she wants and request it for a certain period of time. The owner of the car receives a message and can confirm or decline the request. Other functions of the application is a map view on which the available cars can be displayed and directly booked by the customer. There are also options to list all requests or orders in the app. In the next release, among other things, a chat functionality will be implemented. The goal of the application is to provide a cheap alternative for customers compared to other car sharing providers and the function to rent out his private car and get some money for it. In addition, the application offers a contribution to the environment since customers do not have to buy a new car but can already use an existing one.

1. Introduction

What is CarBNB? CarBNB is a car rental web/mobile app. Do you own one or more cars and do not use them permanently or are you looking for a car to rent? Then this app is just right for you.

As private car owner you can register your car(s) and make them available to other users of the app when you are not using them. In this way, you can earn money with your car(s) during these times instead of losing money.

Are you going on a city trip, a round trip or similar and need or prefer individual transport with a car? Or do you need a car for excursions or shopping at certain times? Then register as a user on CarBnB and you will surely find a good offer to rent a car for your specific purpose.

Of course, you can be in both usage roles, i.e., in the role of offering your own car(s) for rent and in the role of renting cars.

Consequently, there are roles for users: offering your own car(s) for rent or looking for a car for rent. And the user of the app can be in both roles, of course.

This results in various use cases, which will be explained in more detail later. The user must register in order to use the application. In order to offer or rent cars, the user must be logged in. The application offers various search functions, so that you can find what you are looking for. This will be explained in more detail later, as well.

1.1 main features are:

- User registration
- User login
- Register a new car which can be rented by others
- Searching filter for cars depending on different options
- list of all cars which can be rented
- Create a rental
- Approval system that allows future schedule orders
- FAQ section that addresses most important user inquires
- Implemented Tokens authentication and encrypted passwords
- Ability to hide/show your cars for rentals
- Map display of cars locations
- Communication between the owner of car and the user who wants to rent the car (still open)

1.2 Motivation

Most people like to travel. Especially short city trips are very popular. But many people want to remain independent of public transport. Commercial car rentals are expensive though. So, what could be more obvious than saving money with an application like CarBNB.

Of course, a platform like ours also needs customers who offer their private cars for rent. Most people in Europe prefer private transport, i.e., owning their own car. But this car is not used 365 days a year. On these unused days it could be made available for rental.

So, there is enough potential for customers for our application.

1.3 Target Audience

The target group of our application are divided into two groups. First the customers who want to rent a car and second the customers who want to offer a car for rent.

Almost everyone owns a smartphone these days and it’s so easy to offer or book a car with a smartphone over the internet. With a view clicks you can earn money as a car owner. And as a customer who wants to rent a car, you can save money and even do something for the environment.

1.4 Reasons to Use Our CarBNB Application

But why should you, as a car owner, think about using this application and offer your car(s) for rent? Even if your car is standing still and not being used, it loses value. However, if you make your car available to other, you can get compensate for this loss in value und even make a profit.

And why should you, as a user who wants to rent a car, use this app? Because it is cheaper than a normal car rental and more environmentally friendly. Since the cars were not specially purchased for this purpose.

2. Field Overview and Best Practice

2.1 Why Technologies Were Chosen

The technology stack of our project consists of Flutter as the frontend framework, Node.js with Express as the backend technology, and MySQL as database.

The decision to use Node.js as the backend framework was easy because most of the team members already had some experience with it from previous projects. Developing REST APIs with the Express framework is quite easy. Also, there are many Node.js modules that can be used to add additional functionality (e.g. JWT tokens, encryption, ...) without having to write much code. In addition, there are a lot of tutorials and documentation for Node.js, so it's very easy to get started and you can get a lot of help online.

For the frontend, we decided to go with Flutter because we wanted to create an App and access the native features of the underlying system. This was because we wanted to implement things like GEO location features for finding cars. We also wanted to support as many systems as possible without needing a separate code base. That's why we decided against a native app.

For the database, we first had to decide whether we wanted to use a relational or a non-relational database. Since we wanted to have a fixed schema in our database, we decided to choose a relational approach. Since some team members already had experience with MySQL from previous projects, we decided to go with this approach.

2.2 What Could Have Been Done Better

Looking back, we would say that the decision to use this technology stack was a very good one. The implementation on the backend side was easy and worked very well. Implementing the frontend was a bit more difficult because the team members didn't have much experience with Flutter and it takes some getting used to when you start using Flutter and Dart. But in the end it worked also quite well because we helped each other when programming in the frontend. And since Flutter was the best fit for our needs, we would say that it was the right decision. Also, we learned a lot through this approach

3. Goals

We setup goals to achieve in this project, in order to have interesting competitive functions, and to build our knowledge as a team, while also meet the deadline.

3.1 Initial goals

Those were the functions that we set and started with:

- Register a new car which can be rented
- List of all cars which can be rented
- Filter the cars depending on different options
- Create a rental
- Contact the owner of the car
- Build application that have strong backend and

All the mentioned goals were achieved expect contact the owner of the car.

3.2 Added function

Those were the function that we found it will work better and offer more practical and necessary functions:

- Approval system that allows future schedule orders
- FAQ section that address most important user inquiries
- Implemented Tokens authentication and encrypted passwords
- Added ability to hide/show your cars for rentals
- Added map display of cars locations

All the mentioned functions were implemented successfully.

3.3 Future functions

Those are functions that we thought about that we could add later to further improve the app and make it ready for market:

- Include messenger function inside the app
- Adding user registration, Driver license verification
- Adding user profile functions (picture, edit info and password reset)
- Adding multiple photos slideshow for cars
- Adding option to have the car picked up or delivered to you by a fee placed.
- Add gas tank agreement when renting the car
- Add ability to export a report of the orders history

4. Approach and Solution Design

4.1 High Level Architecture Design

4.1.1 Client-server paradigm:

Most modern applications use a client-server paradigm which is also best suited for the system we wanted to develop. A client-server paradigm and application constitute logical layers that can be divided amongst the client and server. For Carbnb we found the two-tier client-server model to be a good fit since we intended to keep the client-side lightweight to optimize for quick page loading and smooth user interaction. To ensure the server-side could do the heavy lifting we researched for the most robust server-side technologies in use today and shortlisted a few. As shown in Figure 4.1

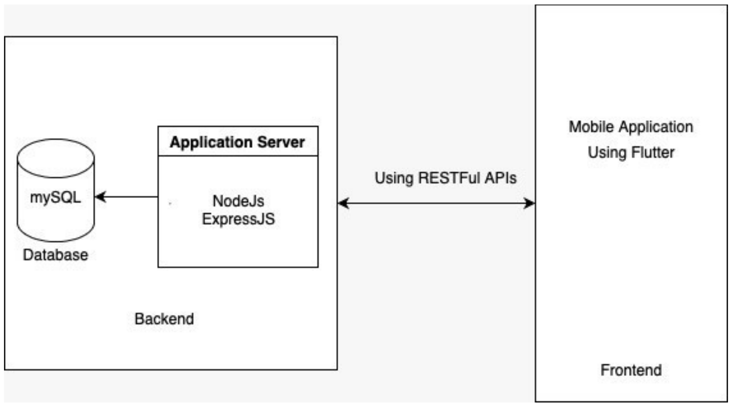


Figure 4.1: High Level Architecture Design

We use the classical client-server architecture for this. On the one side, we deploy nodeJS server and MySQL database on our local machine and we build our frontend app on the client-side which can access using Restful APIs as you can observe in the diagram above.

4.1.2 Authorization and encryption:

Security is one of the most important aspects of software development. For this purpose, we utilized the best practices of the industry and selected JWT (JSON web tokens) for authorization users' requests. All user-sensitive information such as user's passwords is encrypted and stored on the server.

4.1.3 Communication strategy:

To connect the client and server we decided to use RESTful web services as RESTful services are the standard in modern web development due to their simplicity and lightweight signature and web sockets for implementing the push-based polling.

4.2 Use Cases

4.2.1 Use Case: 1 Registration:

A user wants to rent out any car from our Carbnb app. First, he/she need to download an app on their smart phone, Android/iOS. Then open the app, click on the Registration button, fill the form with First Name, Last Name, Email, Phone (optional), password, and confirm the password and do the Registration.

4.2.2 Use Case: 2 Filter Cars:

A user wants to rent a car but preciously wants a specific car. User opens an app if he/she is logged in user then, he/she can press filter button on top, from where user can select location (city), per day maximum price, car brand, type, color, number of seats, fuel type and car transmission type. After applying all the desired filters, the user can see filtered cars.

4.2.3 Use Case: 3 Car by Location:

A user can check car location to estimate pick-up location. The user opens the app and from the bottom tab bar press on Map screen, on the app user can see the car's pinpoint location. Users can change location(city) from top to check cars available in different cities. If the user scroll on the bottom side of the map, can browse different cars and check their details.

4.2.4 Use Case 4: Rent Car:

If a user wants to rent a car, press the detail button of the car he/she is interested in it. From the detail page press the rent a car button on the bottom, select the time duration of the rental period and press the 'Request This Car' button. It will generate a request to the owner and if the owner accepts the requesting user can have that car. In the meantime, users can apply for multiple cars

4.2.5 Use Case 5: Accept/Reject Requests:

If the user has already uploaded his/her car in Carbnb to rent out. He/she can get multiple offers for his/her multiple cars. To check the request, open the app, from the bottom user can press the Menu tab and My Request option, Where the user can see multiple requests of his/her multiple cars, and he/she can accept or reject any request.

4.2.6 Use Case 6: Upload/Register Your Car:

The user can upload his car details to make it available to rent. User open app, press + button on the top right corner, fill out the form with title, description, price per hour, number plate, city, brand, pick-up/delivery type, car type, color, number of seats, number of doors, fuel types, car transmission and can upload an image of the car and submit the form, all fields are mandatory to fill.

4.3 Database and ERD

In data base we created an ERD first then we generated the SQL script to create it using MySql workbench, and to replicate the database between our team members environment we generated SQL database creation script that include the tables, views and even the data to populate the database. And we shared the script on GitLab in order to executed locally on each machine.

4.3.1 Tables

Table	Description
User	This table is to store the user information like name, email and other details including the user encrypted password.
Car	This table is to store all the owner cars with it’s details and description that are uploaded and can be shown on the website for rental
Order	This table is made for all the rental request that is made in the app on any car available
Insurance	We created this table to store the order insurance selected plan, with it’s detail and price effect.
FAQ	We created this table in order to store the main question on the app dynamically.
OrderStatus	This table is made to specify all possible order status and link them to the orders
Brand	This table include all the possible brands for car in order to have fixed data to avoid typo from the client side and also to use later in filter, plus to avoid repetition in the car table.
Type	This table is also linked to the car table and it’s to specify the car types such as Sport car or SUV.
Media	This table is for linking the car to it’s images, and it created separately in order to support user adding multiple images.
FuelType	This Table is to display the user

4.3.2 ERD

Look at he following Figure 4.2

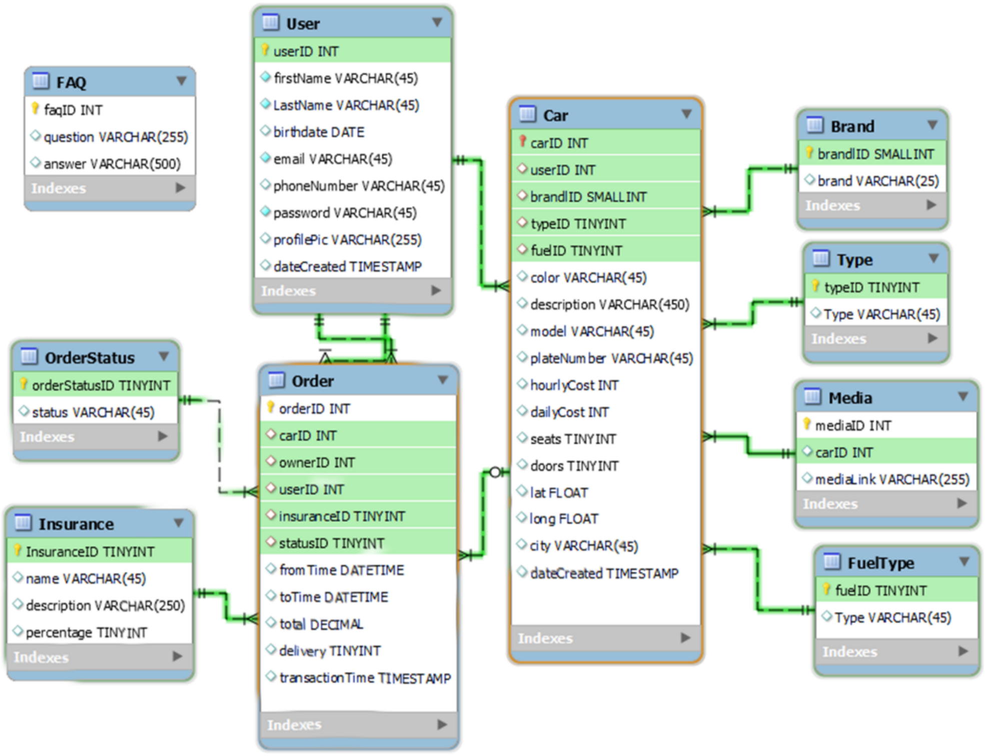


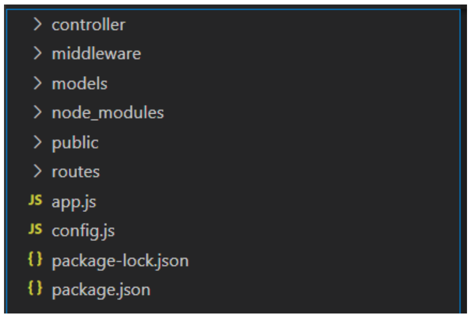
Figure 4.2 CarBnB ERD

4.4 Backend Approach and Design

For the backend, we used Node.js, a runtime environment for implementing server-side logic using JavaScript. In addition, we used the Express web framework to implement the web server functionalities.

4.4.1 Backend structure

The backend code is divided into several separate directories to have a clearer structure in the code. The following screenshot shows the structure of our backend code. Figure 4.3



```
> controller
> middleware
> models
> node_modules
> public
> routes
JS app.js
JS config.js
{} package-lock.json
{} package.json
```

Figure 4.3 Backend structure

The most important parts are the routes, controller, models directories, which will be described in the next sections. The node modules folder contains the external dependencies that we used in the code. These are defined in the package.Json file and are automatically installed with the npm install command. The public folder contains static files, which in our project are the images of the cars.

The graphic in Figure 4.4 shows an overview of the overall backend design.

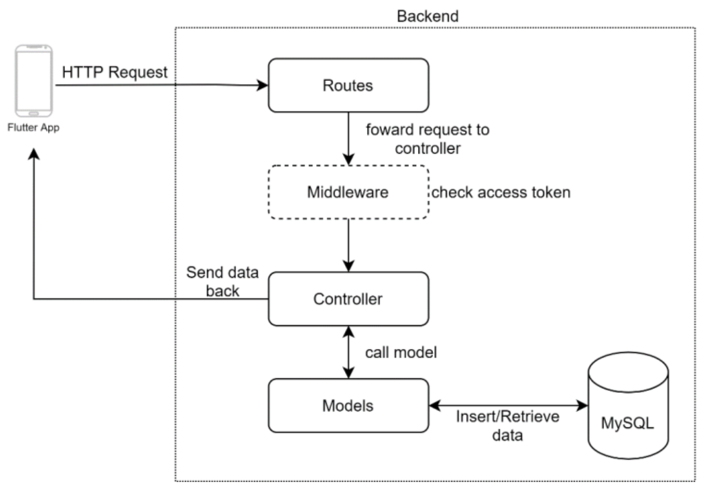
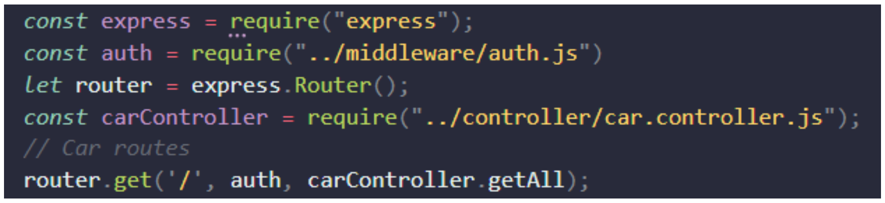


Figure 4.4 backend design

4.4.2 Routes

The routes define the REST API endpoints of the web server. We have used the Router middleware of the Express framework to define these endpoints. The screenshot in Figure 4.5 is from the car routes file and shows an example of how to create such an endpoint.



```
const express = require("express");
const auth = require("../middleware/auth.js")
let router = express.Router();
const carController = require("../controller/car.controller.js");
// Car routes
router.get('/', auth, carController.getAll);
```

Figure 4.5 example of how to create such an endpoint

Each route is defined by an HTTP method and a path, which is the endpoint that the frontend will call. The request is then forwarded to the appropriate controller function. Before that, the request is checked by the auth middleware, which will be described later.

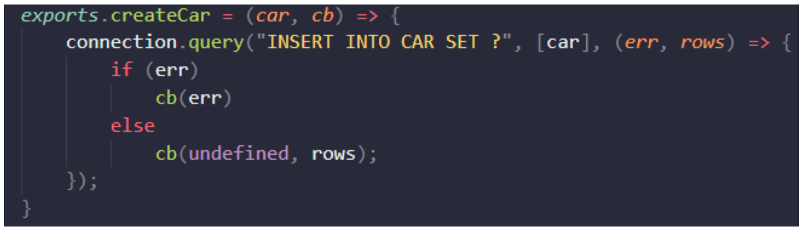
4.4.3 Controller

The controller contains the main logic of the backend. They receive the request and do things like validate the request parameters, call the models for database access, and send data back to the caller, which is our frontend application. In the user controller for example we encrypt the password of the user before it is saved in the database. For the encryption we use the decrypt module.

4.4.4 Models

The models are the components that execute the database queries. We have added the MySQL package to the dependencies of the Node.js application, which is a MySQL driver. This module will handle all the queries. In the models, we define the SQL statements that will then be passed to this module.

The screenshot in Figure 4.6 shows an example.



```
exports.createCar = (car, cb) => {
    connection.query("INSERT INTO CAR SET ?", [car], (err, rows) => {
        if (err)
            cb(err)
        else
            cb(undefined, rows);
    });
}
```

Figure 4.6 Creating a module

We use the question mark to define a placeholder, which is then automatically filled in by the mysql module. The advantage of this syntax is that the mysql module do the escaping of all input parameters. Therefore SQL injections are not possible when using this syntax.

4.4.5 Middleware

The middleware folder contains the auth middleware that checks if a request comes from an authenticated (logged in) user. We used the jsonwebtoken package to implement this. First, the middleware checks if an access token was passed as a bearer token in the authorization header of the request. If so, the jsonwebtoken module verifies the token. If it is a valid token, the request is forwarded to the controller, as shown in the screenshot of the route section. Otherwise, a response with the code 401 (Not authorized) is returned.

4.4.6 API documentation and testing

For testing and documenting the REST APIs, we used the tool Postman. It provides the functionality to make a request to a specific endpoint and also pass parameters and request headers. The tested APIs can then be exported to a JSON file that we have stored in our GitHub repository. The file can then be imported by other team members into their Postman to get the same request with the same parameters.

In addition, we have created a separate documentation with all requests, their parameters and the possible response codes and messages in an AsciiDoc file.

4.5 Frontend Approach and Design

When deciding tech stack for this project we consider all the aspects and difficulties which we might encounter in the future. We want to select a tech stack on which everyone feels comfortable working. Initially, we decide to go with any hybrid solution, not just because of cross-platform functionality but it is also hot in the industry and job market.

For hybrid development there were two frontend approaches which we discuss, one was React-Native, and the other was Flutter. We don't have any issue starting with any of these as both are stable and have industry reorganization. Once we discuss this in our team meeting, we found that some of us already have experience in Flutter while others were total newbies. So, we decide to go with it so we all can improve or learn new Flutter skills.

Flutter is an open-source UI software development kit back by Google. It is used to build cross-platform Web and mobile apps. We this to build or Frontend App Carbnb.

Flutter gives a declarative UI design approach where you handle the states of widgets. The function will build the UI, which describes the state. Therefore, the declarative framework means, it just shows the current state of the widget on the UI. So, this is the concept of Flutter declarative. As this simple example, every other complex changes also happen as same.

We try to use the MVC design pattern for our frontend app architecture where we are using a Model to hold data from API, view is responsible to handle UI-related functionality while the controller is responsible to fetch data from models to view it on views. Along with MVC, we use layered architecture for API calling where we create a base layer of our networking class on which we have a different logical layer, like Login_api_manager, car_api_manager, or order_api_manager.

To handle user different states, we use persistence memory of the mobile device, a kind of local database, in which we stored login user data in hash-map (Key-value pair) data structure which we are retrieving on app opening to check if a user is a login to show appropriate UI. We are also using the same data on my profile screen.

5. Implementation and Prototype

5.1 API's

We created over 43 different Restful API's in order to cover all our use cases and to have ready to implement additional functionality based on the progress we made in the front end.

5.1.1 Cars

Get	**/car**	Get All Cars
	/car/filterdata	Get filters Data
	/car/1	Get Car by ID
	/car/user/2	Get User Cars
Post	**/car/**	Add New Car
	/car/filter	Get filtered result
Put	**/car/**	Update a Car
	/car/carstatus	Update car status
Delete	**/car/2**	Delete a Car

5.1.2 User

Get	**/user/all**	Get all users
	/user/1	Get user by ID
Post	**/user/register**	Register
	/user/login	Login

5.1.3 FAQ

Get	**/faq**	Get all FAQ
	/faq/1	Get FAQ by ID
Post	**/faq**	Add new FAQ
Put	**/faq**	Update FAQ
Delete	**/faq/1**	Delete FAQ

5.1.4 Brand

Get	**/brand**	Get all brand
	/brand/1	Get brand by ID
Post	**/brand**	Add New brand
Put	**/brand**	Update brand
Delete	**/brand/1**	Delete brand

5.1.5 FuelType

Get	**/fueltype**	Get all fuel types
	/fueltype /1	Get fuel type by ID
Post	**/fueltype/**	Add new fuel type
Put	**/fueltype/**	Update a Fuel type
Delete	**/fueltype/2**	Delete a Fuel type

5.1.6 CarType

Get	**/cartype**	Get all car types
	/cartype /1	Get car type by ID
Post	**/cartype/**	Add new car type
Put	**/cartype/**	Update a Car type
Delete	**/cartype/2**	Delete a Car type

5.1.7 Order

Get	**/order**	Get all orders
	/order/2	Get user requests by ID
	/order/pendingOrder/3	Get owner pending order for approval
	/order/ownerOrder/2	Get owner requests by id
Post	**/order**	Make new order
Put	**/order**	Update order status

5.1.8 Insurance

Get	**/insurance**	Get all insurance plans data

5.1.9 Media

Get	**/media/5**	Get all images for a specific car

5.2 Mockups

At the beginning of the project, over 14 mockup screenshots were created to define the design to a large extent before programming started. In the following you can find some screenshots of our mockups. Figure 5.1, 5.2 and 5.3

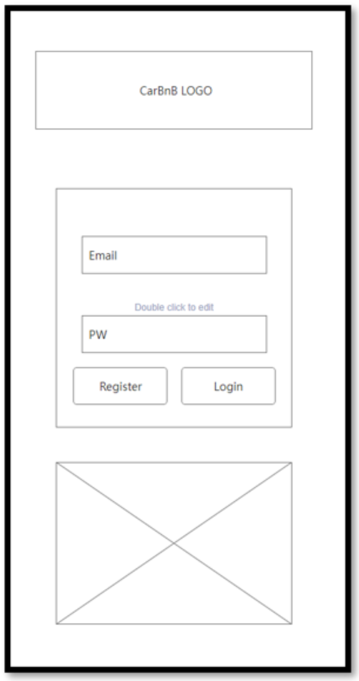


Figure 5.1 Login Screen

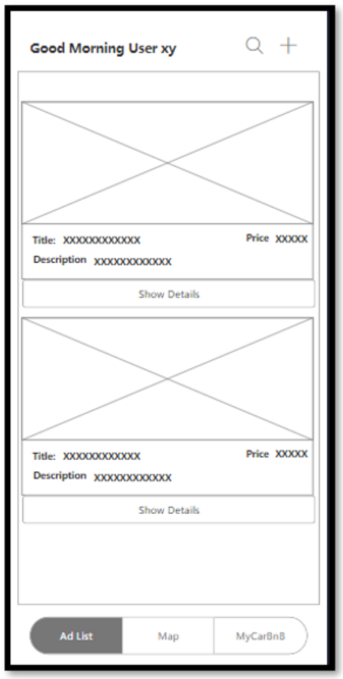


Figure 5.2 Home Screen

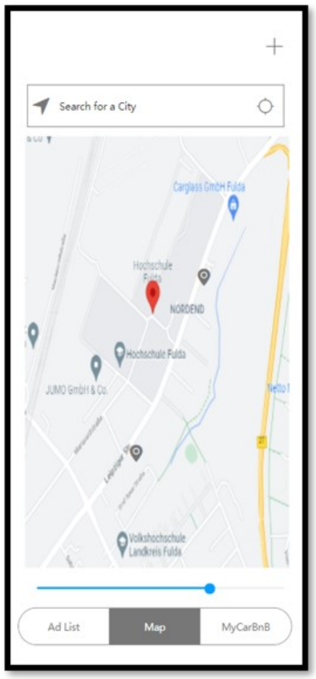


Figure 5.3 Map Screen

5.3 App Screenshots

The figures from 5.3 to Figure 5.15 show the application implemented interface.

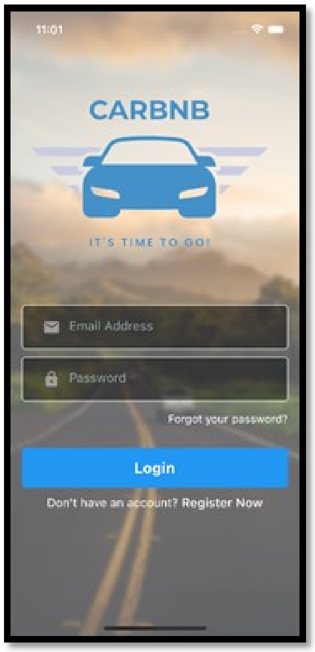
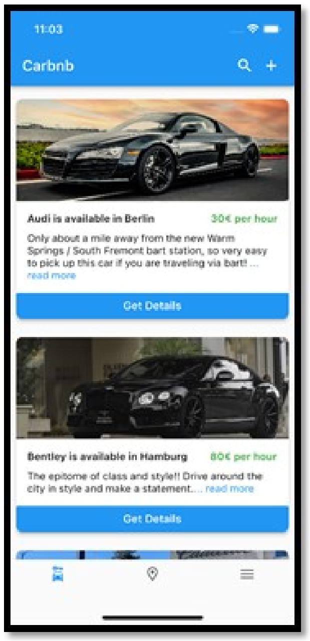


Figure 5.4 Login Screen



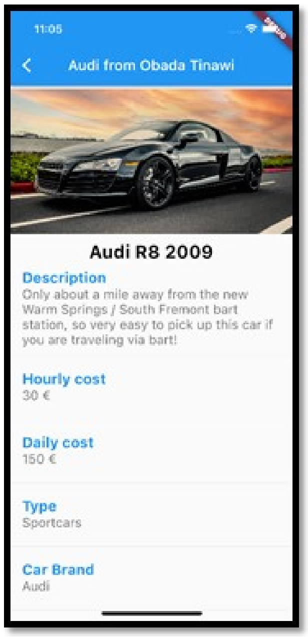
Audi is available in Berlin 30€ per hour

Only about a mile away from the new Warm Springs / South Fremont bart station, so very easy to pick up this car if you are traveling via bart! ... read more

Bentley is available in Hamburg 80€ per hour

The epitome of class and style!! Drive around the city in style and make a statement.... read more

Figure 5.5 Home Screen



Audi R8 2009

Description
Only about a mile away from the new Warm Springs / South Fremont bart station, so very easy to pick up this car if you are traveling via bart!

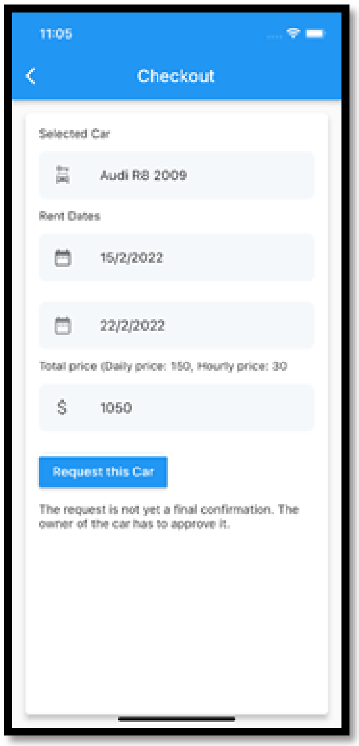
Hourly cost
30 €

Daily cost
150 €

Type
Sportcars

Car Brand
Audi

Figure 5.6 Detail Screen


11:05

Checkout

Selected Car

Audi R8 2009

Rent Dates

15/2/2022

22/2/2022

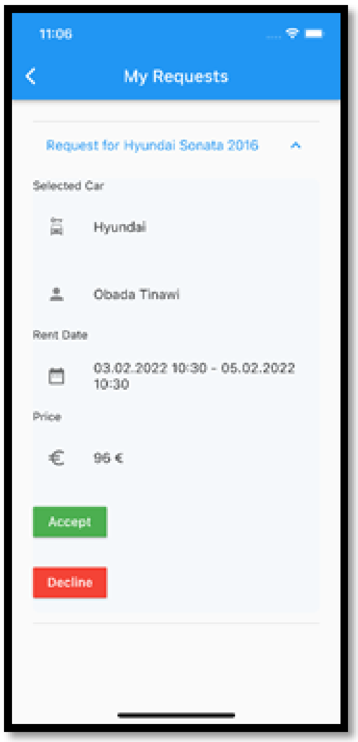
Total price (Daily price: 150, Hourly price: 30

$ 1050

Request this Car

The request is not yet a final confirmation. The owner of the car has to approve it.

Figure 5.7 Checkout


11:06

My Requests

Request for Hyundai Sonata 2016

Selected Car

Hyundai

Obada Tinawi

Rent Date

03.02.2022 10:30 - 05.02.2022 10:30

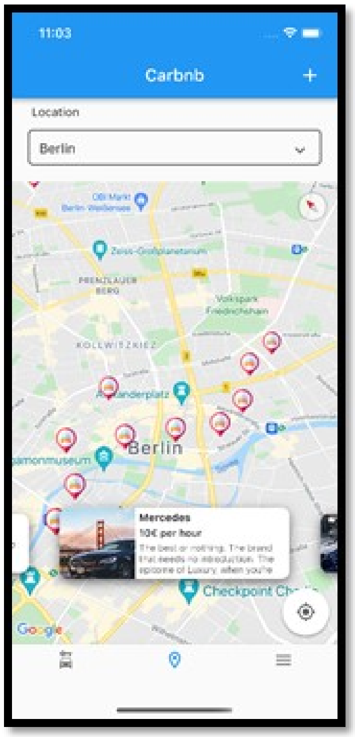
Price

€ 96 €

Accept

Decline

Figure 5.8 My Requests


11:03

Carbnb

Location

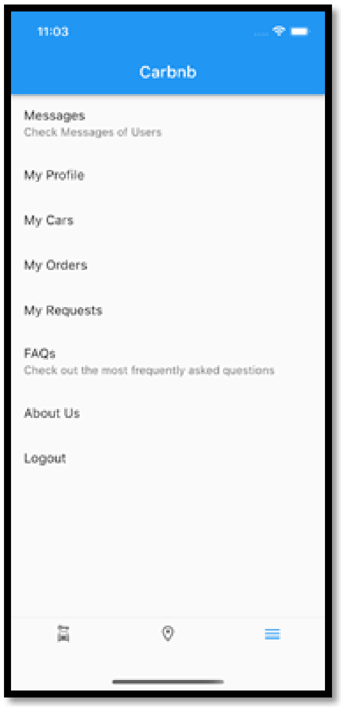
Berlin

Mercedes

10€ per hour

The best or nothing. The brand that needs no introduction. The epitome of Luxury, when you're

Figure 5.9 Map Screen


11:03

Carbnb

Messages

Check Messages of Users

My Profile

My Cars

My Orders

My Requests

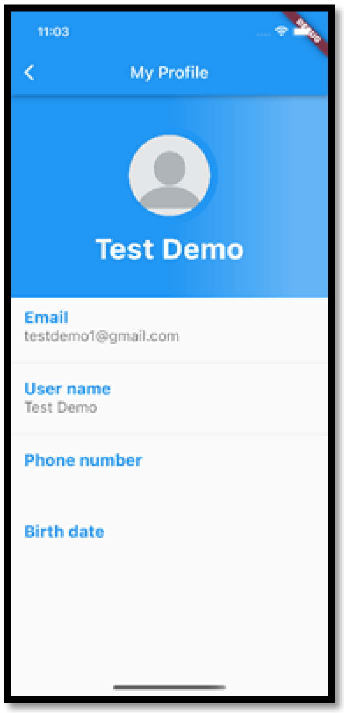
FAQs

Check out the most frequently asked questions

About Us

Logout

Figure 5.10 MyCarBnB


11:03

My Profile

Test Demo

Email

testdemo1@gmail.com

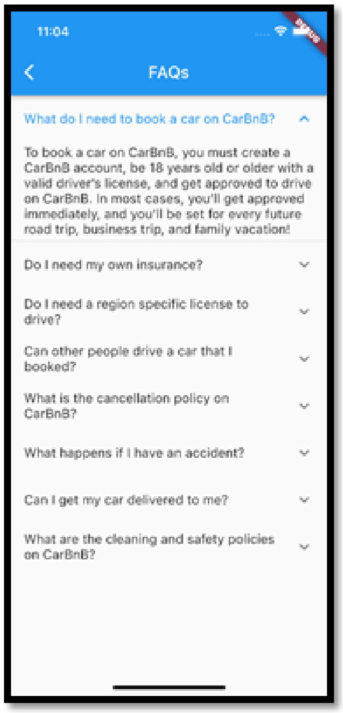
User name

Test Demo

Phone number

Birth date

Figure 1.11 My Profile


11:04

FAQs

What do I need to book a car on CarBnB?

To book a car on CarBnB, you must create a CarBnB account, be 18 years old or older with a valid driver's license, and get approved to drive on CarBnB. In most cases, you'll get approved immediately, and you'll be set for every future road trip, business trip, and family vacation!

Do I need my own insurance?

Do I need a region specific license to drive?

Can other people drive a car that I booked?

What is the cancellation policy on CarBnB?

What happens if I have an accident?

Can I get my car delivered to me?

What are the cleaning and safety policies on CarBnB?

Figure 5.12 FAQs

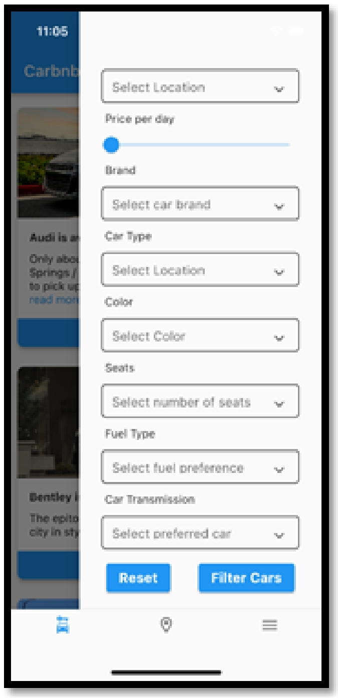


Figure 5.13 Filter Screen

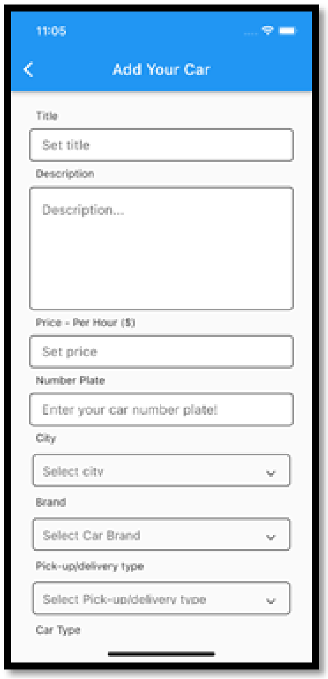
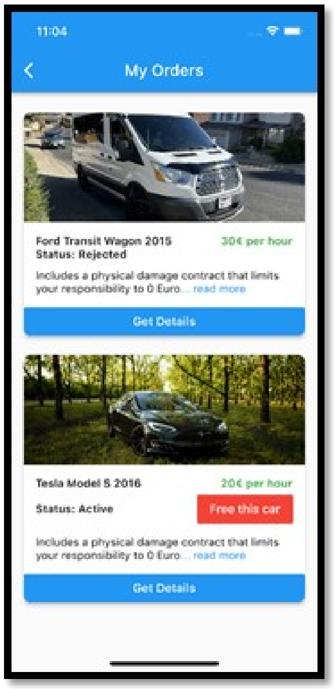


Figure 5.14 Add Your Car



Ford Transit Wagon 2015 30€ per hour
Status: Rejected
Includes a physical damage contract that limits your responsibility to 0 Euro... read more

Tesla Model S 2016 20€ per hour
Status: Active
Includes a physical damage contract that limits your responsibility to 0 Euro... read more

Figure 5.15 My Orders

6. Critical Reflection of Achieved State

Looking back, Team 2A draws a positive reflection of this project. In the beginning, the learning curve regarding the technological stack was very high. The team members had to work out and familiarize themselves with the programming parcels. Members who had questions were supported so that they could continue working quickly. This way almost all "must-have" goals from the pitch were achieved. Due to time constraints, the messaging feature could not yet be implemented in this release. However, this is planned for a future version. In addition, all "Nice-to-Have" goals were achieved. Even functions that were not on the scope at the beginning, such as a map function, were implemented.

There were no major conflicts in the team except for one point. One team member had temporarily left the team for various reasons for some weeks and the rest of the team had to redistribute the tasks from this person. However, the team member decided to rejoin for the documentation part, and she was reintegrated into the group. Also, organizational things like meetings or short-term calls of the team members were no problem. In the future, we would perhaps pay a little more attention to time management, so that all to-dos are completed on time.

7. Summary and Outlook

7.1 Summary

This project journey was amazing for us, we started by creating a team and establish a line of communication on MS-teams, then get to know each others and find the pest technology to use in order to utilize our past experiences. And searching and voting to have the best idea to implement was also very rewarding.

We made discussion to better shape the idea, and after we introduced our idea to the professor and classmates we got very useful feedback that play huge role in our core functions.

We split the work between our member evenly and once a member finish earlier, they rush to do other stuff for the project.

We made weekly meeting to discuss every step we did and further polish it with others feedback.

And after we started programming the communication between frontend and backend was flawless, and we were able to quickly identify requirements and create it from both sides efficiently.

And we successfully were able to meet our deadline and deliver a working and functional app MyCarBnB that allow any car owner to offer his car easily for rental and for any user to easily rent it. This project was a huge learning opportunity for building knowledge, working in teams, implementing feedback, presenting and share progress with other classmates.

We are proud that our project achieved the desired outcome and won the award of the best project in the class, it’s a huge owner for us.

7.2 Outlook

Most of our “Must-Have”-Features have been accomplished with one exception: We wanted to implement the feature to contact the owner of the car. In a future version this function will be implemented, too.

Also, we have plans for some new innovative features for the release 2.0.

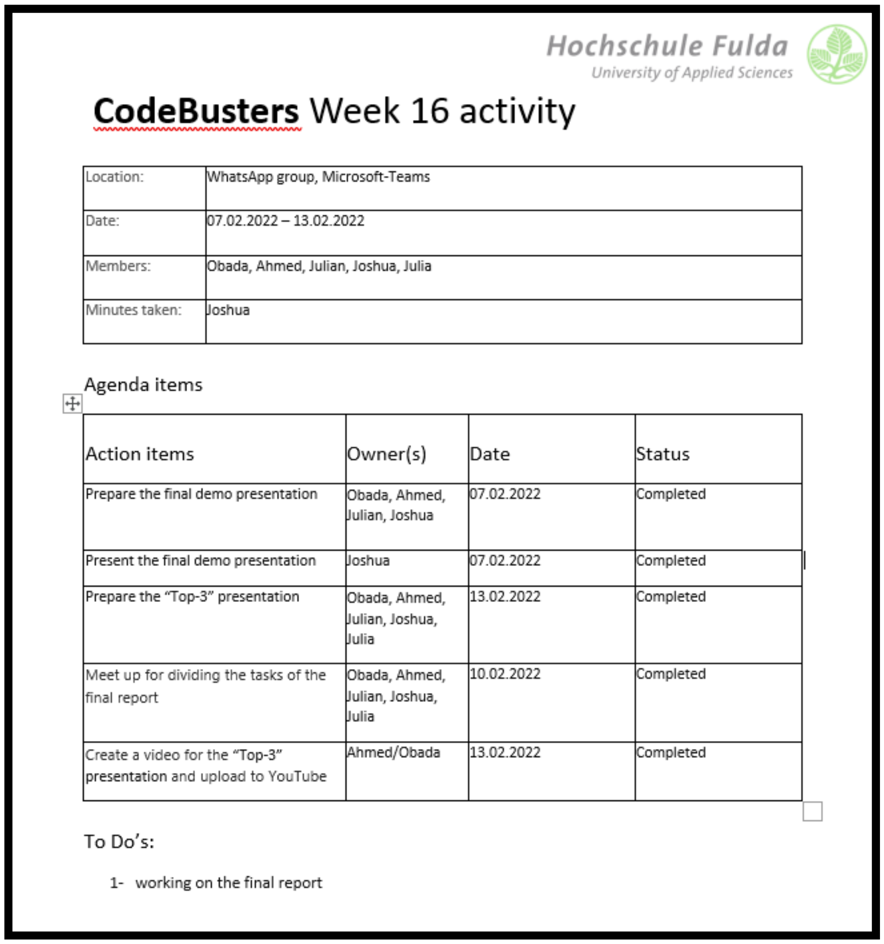
7.2.1 *New features for release 2.0 / TODOs*

- Messenger function inside the application, so that customers can stay in touch, if there any questions about the rental or the car.
- User profile. The user of the application can upload a user picture, write and edit user profile information and manage the password settings
- Possibility to create a slideshow for the cars which the user offers for rental.
- Add an option that the car can be delivered or picked up to you for a fee.
- Include an option to select a gas tank agreement by the user when renting a car.
- Including also the ability to rent cars on hourly basis. Some customers need a car only for a short period and short distances. With this new feature, the can find what they in need for.

8. Appendix

8.1 Protocols

We usually met 2 times a week and discussed open tasks and set new goals. This was recorded in meeting minutes. Here is an example of one, the other of the meeting minutes can be found in Gitlab. Figure 8.1



Hochschule Fulda
University of Applied Sciences

CodeBusters Week 16 activity

Location:	WhatsApp group, Microsoft-Teams
Date:	07.02.2022 – 13.02.2022
Members:	Obada, Ahmed, Julian, Joshua, Julia
Minutes taken:	Joshua

Agenda items

Action items	Owner(s)	Date	Status
Prepare the final demo presentation	Obada, Ahmed, Julian, Joshua	07.02.2022	Completed
Present the final demo presentation	Joshua	07.02.2022	Completed
Prepare the "Top-3" presentation	Obada, Ahmed, Julian, Joshua, Julia	13.02.2022	Completed
Meet up for dividing the tasks of the final report	Obada, Ahmed, Julian, Joshua, Julia	10.02.2022	Completed
Create a video for the "Top-3" presentation and upload to YouTube	Ahmed/Obada	13.02.2022	Completed

To Do's:

1- working on the final report

Figure 8.1 Protocol Example

8.2 Setup to run "MyCarBnB"

To make our application work, some requirements are necessary, which are briefly explained in the following part.

8.2.1 *The following programs/frameworks must be installed:*

- Flutter
- Android Studio
- NodeJS
- MySQL + MySQL Workbench
- GIT

8.2.2 *Instruction:*

- Clone the Gitlab-Repository into an IDE (e.g. Android Studio):
- Link: https://gitlab.cs.hs-fulda.de/fdai5232/mobile-apps-group-2a.git
- Execute the SQL-Script (V2), which is inside the Repository in a MySQL Database
- Switch to the Project folder (mobile-apps-group-2a → backend) and type in terminal "npm start"
- In Android Studio the application can be started via a emulator/web browser
- To use the application, you can register as new user and experience our App

9. Tasks Per Team Member

9.1 Report Part

Part	Student
1-Title with product name, Document formatting	Obada Altinawi
2-Abstract	Joshua Rahimi
3-Introduction/motivation	Julia Nürnberg
4-Field overview	Julian Günther
5-Goals	Obada Altinawi
6-Approach / Solution design	
• High level Architecture Design	Ahmed Shahid
• Use cases	Ahmed Shahid
• Database and ERD	Obada Altinawi
• Backend approach and design	Julian Günther
• Frontend approach and design	Ahmed Shahid
7-Implementation / Prototype	
• API's	Obada Altinawi
• Mockups	Joshua Rahimi
• App screenshots	Joshua Rahimi
8-Critical reflection of achieved state	Joshua Rahimi
9-Summary	Obada Altinawi
10-Outlook	Julia Nürnberg
11-Appendix	Joshua Rahimi
12-Tasks per team member	Obada Altinawi
13-References	All Members

9.2 Coding part

Part	Student
Creating GitLab repository	Julian Günther
Initializing the project node.js and flutter	Ahmed Shahid
Creating ERD and database and populate it.	Obada Altinawi
Adding routes, controllers, models to backend	Julian Günther
Creating the Mockups	Joshua Rahimi 80%
	Ahmed Shahid 20%
Creating backend API's	Julian Günther 40%
	Obada Altinawi 60%
Creating Frontend screens	Ahmed Shahid 50%
	Joshua Rahimi 40%
	Julian Günther 10%
Creating Frontend functions	Ahmed Shahid 50%
	Joshua Rahimi 20%
	Julian Günther 30%

10. References

1- Express/Node introduction
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
2- NodeJS + Express part 5: Routes and Controllers
https://dev.to/ericchapman/nodejs-express-part-5-routes-and-controllers-55d3
3- The software used to make mockups
https://www.mockplus.com/
4- Flutter guide
https://docs.flutter.dev/
5- debugging and asking the open source community using
https://stackoverflow.com/