Technical Report - **Product specification**

Car Management Application

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 07, October, 2024
Students:	114646: Francisco Albergaria 113682: Gabriel Santos 113207: Guilherme Amaral 114514: João Gaspar
Project abstract:	This project aims to create a Car Management App that uses information retrieved from the ECU (Electronics Control Unit) of a car to display real-time warnings and details about it.

Table of contents:

1 Introduction

2 Product concept
- Vision statement
- Personas
- Main scenarios

3 Architecture notebook
- Key requirements and constrains
- Architetural view
- Module interactions

4 Information perspetive

5 References and resources

1 Introduction

In the ambit of Introduction to Software Engineering, we (assigned group 204) compromised ourselves to work as a team, develop code, solve tasks and deliver results according to a normal workplace flow expectation. As so we plan to learn from everything this class has to offer, and to follow the templates, ideas and apply the concepts learnt during the practical and theoretical classes into our project, in which we conceptualize and implement a solution to a Car Management Application.

Our group is constituted of four members, in which we all know each other and therefore already have developed a good team spirit, communicative and critical, meaning we won't hold back onto debating and finding solutions to possible problems during this class.

- Francisco Albergaria – Product Owner
- Gabriel Santos – DevOps
- Guilherme Amaral – Team Manager
- João Gaspar – Architect

Some of the goals in mind for this project, in relation to the scope of the IES course is to understand the organization of a software project, managed as an industrial process, select the best software architecture for a given problem/product and build a software system as a team, using a business framework, while using corporate solutions and tools for software development.

Contents to learn and apply: (Software engineering principles; Software process; Software architectures, Cloud models; Development environments; Spring framework and spring boot; Software certifications; Business and ethics).

2 Product concept

Vision statement

The product envisioned is an application (web/mobile) with the objective of providing valuable information about its user's car in and efficient, concise and simple way. To retrieve such information a custom device, integrated with GPS, is installed in the car, being connected to the ECU (Electronic Control Unit -> gives information regarding everything about the car, such as the stats and status of each component).

With the information provided by the application the user's experience of owning a car would be much facilitated, with the opportunity to get warned about car malfunctions, oil level, breaks/tires states, inspection warning (data both provided by the user so that he can follow it and save it in one place and by the car sensors).

Most modern cars already provide a lot of useful information, for example recent BMW models, which are full of notifications about any problem or details about the car

without the need to be turning on the ignition, watching how many times the motor light blinks and then search in the manual what's the issue. That is what we are aiming for, while also providing the greatest number of details possible, bringing it to all current cars. In that way our app would be able to have the same utility of OBD2 devices, while needing only one installation of the device and providing more information and alerts in a more readable way.

Conclusively, our objective is to make the use of systems like OBD2 more beginner friendly, turning people's attention more towards the importance of having a car well taken care of, since that's the first step for road safety and allowing any entry level user from taking advantage of our app, independently from it being an expert in mechanics or someone who's ignorant of any car concepts. We plan to provide important information that could be used for good, and in an accessible way.

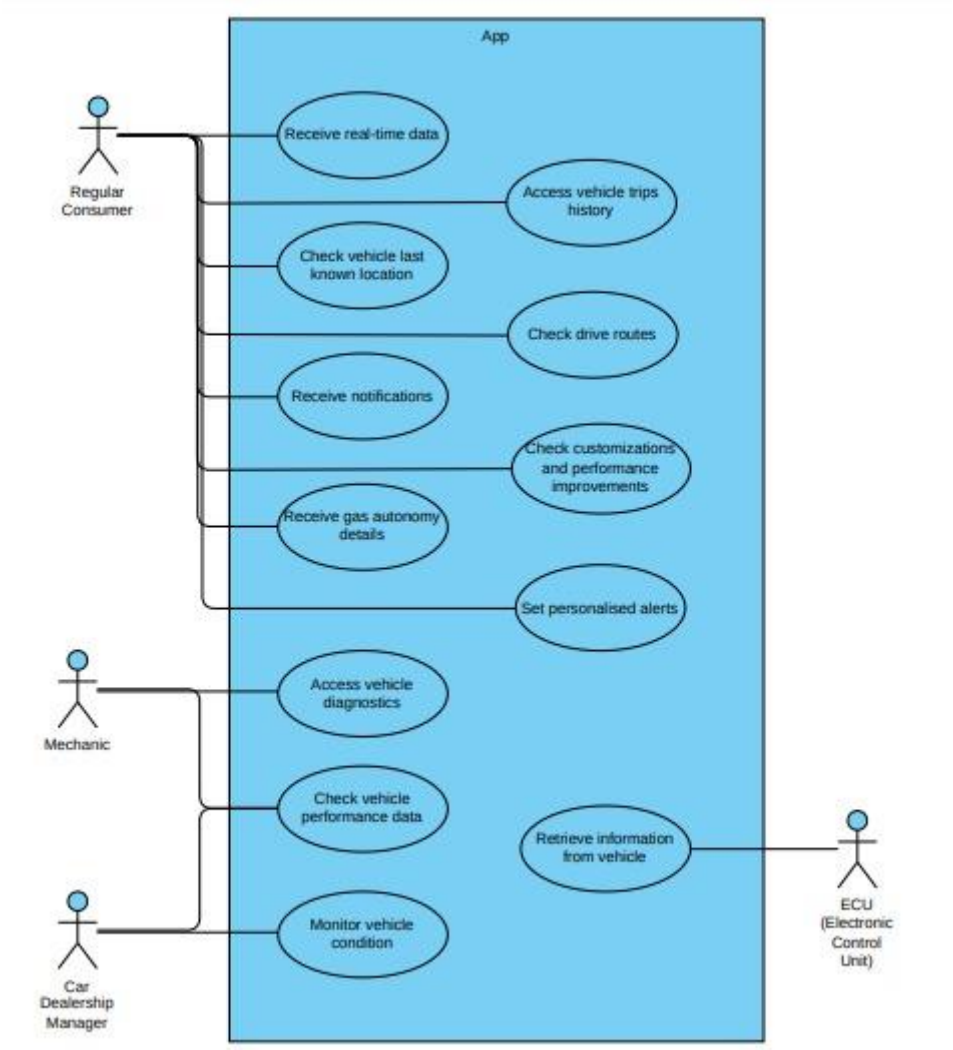

Figure 1. Use cases diagram

The requirements retrieval was solely made by the members of the group, in which the idea was confirmed by the teacher. For us to accomplish this we used predefined ideas and common sense along with research in different websites related to the functioning of

cars as well as the technologies included in one, and the importance of them. (Research links will be posteriorly pointed out in point 5)

Inspiration

As all the members of the group are car users, the concept of the application came from certain needs that we feel that would be fulfilled with certain features, or in this case with our application. Some of us are curious, others have family members who are mechanics, therefore building requirements was facilitated. Besides, rational thinking was used in requirements, along with overall knowledge and searching.

Personas and Scenarios

Personas (Name, Age, Role/Job):

- Maria, 38 years old, Lawyer
- Ricardo, 52 years old, Taxi Driver
- Albano, 35 years old, Teacher
- Inês, 30 years old, Photographer
- Marta, 37 years old, Journalist
- Diogo, 28 years old, Programmer
- Sofia, 33 years old, Race Driving Technician
- Luís, 27 years old, Car Enthusiast
- Hugo, 37 years old, Test Drive Supervisor
- Mateus, 42 years old, Dealership Manager
- António, 58 years old, Mechanic
- Francisco, 30 years old, Musician

Main Scenarios:

- While working at his shop, António, finds that the OBD2 that him and his employees use has a weak display of information and is often confusing to new workers, so he needs a quick and simple way to get all a client's car information. For that he connects our device to the car ECU and, using our app logs in and links the car to his account.
- Marta has a busy life as a journalist, living on the road, and so, already had problems because of her distraction, causing her to forget to take her car to the maintenance and repair. Because of that, she dreams of a way to be warned when she needs to do such things or when something is wrong with her car. To do so she gets our device installed on her car and, using or app, logs in, links her car to her account and starts getting alerts when something is wrong, or maintenance is appropriate.

Product requirements (User stories)

(The personas used are the ones defined in the previous tab)

User Story: As António, I want to access detailed diagnostics of customer vehicles, so I can quickly identify issues and carry out repairs efficiently.

Acceptance Criteria:

- The system provides a clear dashboard showing all vehicle diagnostics in real-time.
- The diagnostics include detailed information such as engine status, battery level, and error codes.
- The system allows filtering diagnostics by vehicle components (engine, transmission, etc.).
- The system allows the link of several vehicles to the same customer.

Priority: 5
Difficulty: 4

User Story: As Hugo, I want to check the last test drive routes and performance data for cars, to ensure that the vehicles are in good condition before showing them to potential buyers.

Acceptance Criteria:

- The system logs all trip routes with timestamps.
- Performance data (e.g., speed, fuel consumption, engine performance) is available for each trip.
- Alerts are generated if any performance anomalies are detected during drives.

Priority: 3
Difficulty: 4

User Story: As Luís, I want to track all the custom modifications and performance improvements made to my car, so I can optimize it for future tuning and events.

Acceptance Criteria:

- The system allows to log modifications made to the vehicle.
- Performance improvements are tracked with before-and-after data (e.g., horsepower, torque, etc.).
- A summary report of all custom modifications (and their impact on performance) is

available.

Priority: 1
Difficulty: 3

User Story: As Mateus, I want to monitor the condition of cars in my dealership, so I can ensure they are maintained properly and ready for sale or test drives.

Acceptance Criteria:

- The system shows the maintenance status of all vehicles associated with the client.
- It alerts the user when any vehicle requires servicing or inspection.
- Vehicle history, including past services and test drives, is available in the system.
- Notifications are triggered if a car has been idle/stopped for a long period.

Priority: 4
Difficulty: 3

User Story: As Sofia, I want to receive real-time data on the car's technical performance during races, so I can adjust and ensure the car is running at its optimal level.

Acceptance Criteria:

- The system provides real-time telemetry data during trips (e.g., engine temperature, tire pressure, speed, etc).
- Alerts are triggered when critical performance thresholds are exceeded (e.g., overheating).
- The user can view past trips data for comparison and optimization.

Priority: 4
Difficulty: 5

User Story: As Diogo, I want to receive alerts for oil and coolant levels, so I can avoid issues and plan maintenance in advance.

Acceptance Criteria:

- Notifications are sent when the oil or coolant level falls below the set threshold.
- Historical data on oil and coolant levels is available for monitoring trends.

Priority: 4

Difficulty: 3

User Story: As Marta, I want to be notified when my car's mandatory inspection is approaching, to avoid fines and legal issues.

Acceptance Criteria:

- The system sends reminders 30 days, 7 days, and 1 day before the mandatory inspection date.
- A countdown to the inspection date is displayed on the dashboard.
- Notifications include required documents and checks for the inspection

Priority: 2
Difficulty: 2

User Story: As Inês, I want to get a prediction of how long my car can run with the current fuel level, so I can plan my trips as a freelancer.

Acceptance Criteria:

- The system calculates an estimate of the remaining driving range based on current fuel levels and recent driving habits.
- Notifications are triggered when the fuel level is critically low.

Priority: 3
Difficulty: 3

User Story: As Albano, I want to check the last known location of my car, so I can remember where I parked it when I forget.

Acceptance Criteria:

- The system shows the last recorded location of the car on a map.
- The location data includes a timestamp of when the car was last parked.

Priority: 2
Difficulty: 2

User Story: As Ricardo, I want to access my car's average monthly fuel consumption, so I can better manage my expenses and earnings.

Acceptance Criteria:

- The system calculates and displays the average monthly fuel consumption in litres/gallons and cost.
- Historical fuel consumption data is available for up to 12 months.

Priority: 4
Difficulty: 2

User Story: As Maria, I want to receive a notification whenever my car is moved, to ensure it hasn't been stolen.

Acceptance Criteria:

- The system sends an instant notification to the phone when the car is turned on.
- The notification includes the time and current location of the vehicle.
- A confirmation request is sent to the user to verify whether the movement was authorized, if unauthorized, the system provides an option to alert local authorities.

Priority: 4
Difficulty: 3

User Story: As Francisco, I want to get the location of close mechanics along with my alerts so that I know where to take my car to fix the issue.

Acceptance Criteria:

- The system gives a list of close mechanics or shops along with the alert every time the car has a big problem
- The list has the GPS location associated with each shop

Priority: 2
Difficulty: 5

Functional Requirements:

- Display of car mechanical information
- Logging System
- Display of car electrical information
- Display of car legal information
- Alert notifications
- Access to car last known location
- Access to recent trips information

- Real-Time information
- Allow multiple cars per user
- Simple device linking process
- Display POI locations

Non-Functional Requirements:

- Secure Data Base
- Concise and consistent Data Base
- Optimized data processing to guarantee the shown data is as close to real time as possible
- Usable on both mobile devices and computers
- User interface must be easy to use
- Be able to handle a broad number of users as well as a large influx of users
- Secure message queue
- System should be maintainable and backwards compatible in case of hardware modifications

3 Architecture notebook

Key requirements and constrains

Considering the above-mentioned requirements and our project goal, it is of high importance for us to think about the following issues and constrains regarding our project architecture:

- There may be different protocols for different car manufacturers so, as such, the system should be able to translate (with an adapter) the data received to a standard format.
- The system will be mainly offered for mobile devices through web app. However, it should be usable on different devices such as tablets and computers.
- The interface must be accessible and easy to use.
- The system must have a history of trips for all users. This information must be securely safeguarded.
- The system must have an authentication system.
- The app can be used at the same time as the device is collecting data, as such, the system should be efficient in gathering, processing and showing the collected data on the app.

Architectural view

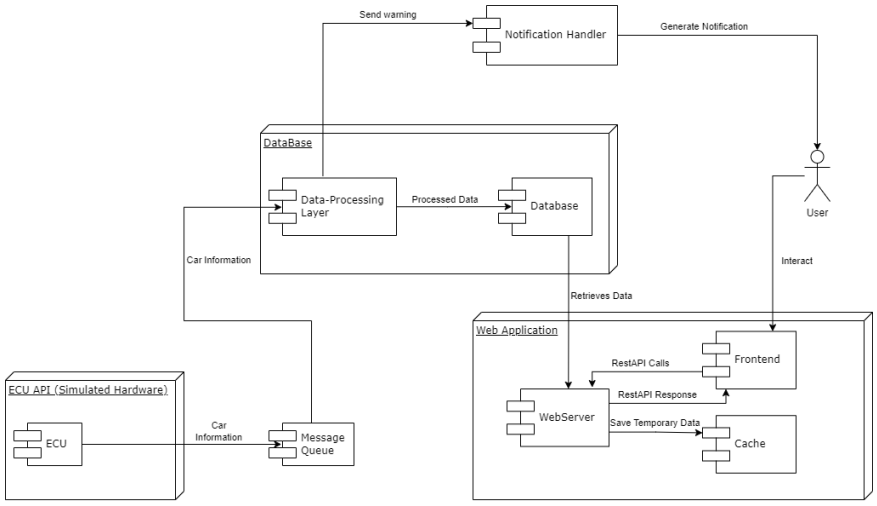

Figure 2. Architectural top view

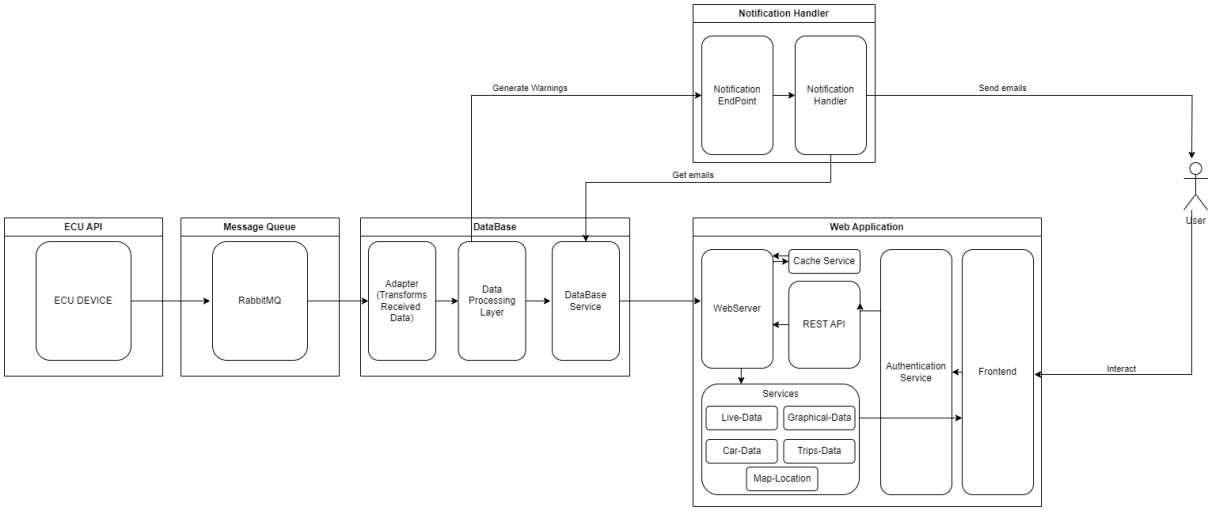

Figure 3. Detailed Architectural view

Data Sources:

- **ECU:** Sends all the car data retrieved to the server by a Message Queue. For this project, the data will be simulated.
- **Message Queue:** RabbitMQ will be used to transfer the data.

Backend:

- **Data-processing Layer**: Gather messages from MQ and processes it, sending the processed data to the database. If a warning threshold is surpassed by the received data, a warning is sent to the notification handler.
- **Database**: All the necessary data will be stored using a MongoDB.

- **WebServer**: Serves as a controller to provide the necessary information to the Frontend through API calls. It gets the information from the database. SpringBoot will be used.
- **Notification Handler**: Guarantees notifications are sent to the users (only necessary for push notifications)

Frontend:

- **Frontend:** Presentation layer, developed in React, Vite and TypeScript.
- **Cache:** Stores session information using Redis.

Module interactions

The way all the modules previously introduced will interact with each other is depicted in the following diagram:

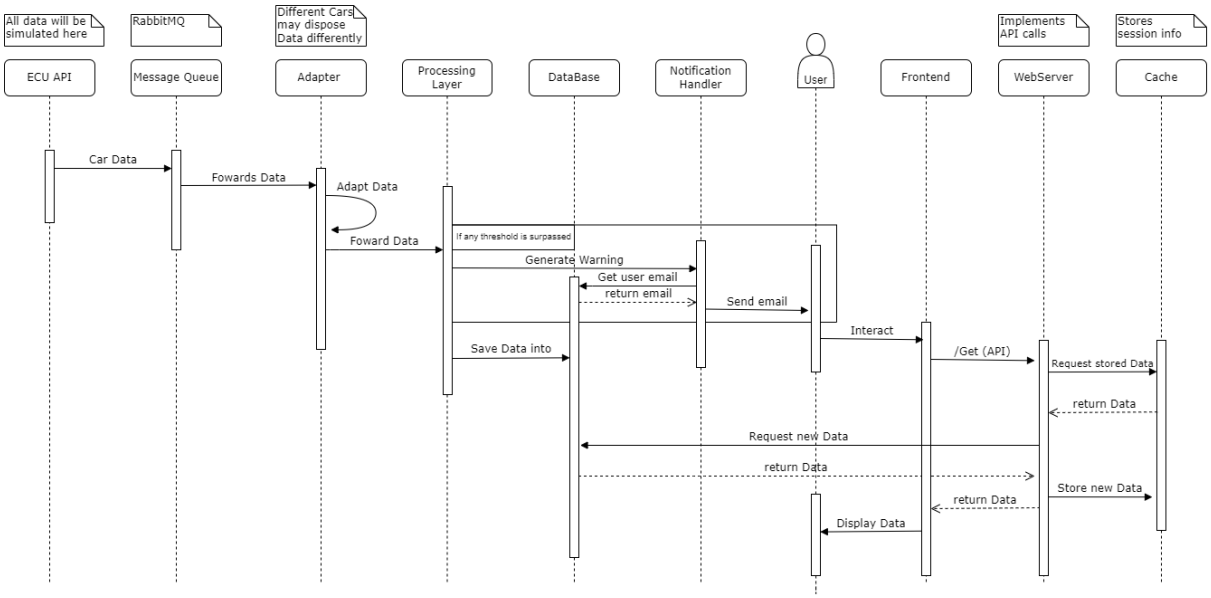

Figure 4. Sequence Diagram

As you can see, our main data source is retrieved from the car ECU, from which is sent, using a **Message Queue**, to an **Adapter** integrated inside our **Data-Processing Layer**. When an anomaly, such as low oil level, is detected when processing the data, a warning is sent to our **Notification Handler** that following the warning gets the email of all users to which the car ECU ID is associated with from the **Database**, sending them an Alert.

In a normal Data flow, after being processed, the data is stored in the **Database**. Every time a user uses our app several requests are made to the **Webserver** by an API. The **Webserver** then retrieves information from both the **Database** and the current session **Cache**. To make our response time faster and not overload the **Database** with requests, when the user is accessing real-time data, our application will store past requests data temporarily in the **Cache**, making it so that it only needs to retrieve the more recent and new data from the **Database**.

4 Information perspective

User: For each user there will be an email, password and list of cars associated to him. Each associated car will be identified by its ECU identifier.

Car: Regards all the information regarding the car's components, including brand, model and engine types. Also saves processed information such as the current autonomy (which is calculated by the last received gas level)

Car_Live_Info: Stores all the information regarding the car's current system status, if it's on, how much battery it has left, etc. This information is later processed to determine autonomy, efficiency and similar.

Trip_Info: Stores all the information regarding each trip's start and finish. Each **Car_Live_Info** is connected to a certain trip.

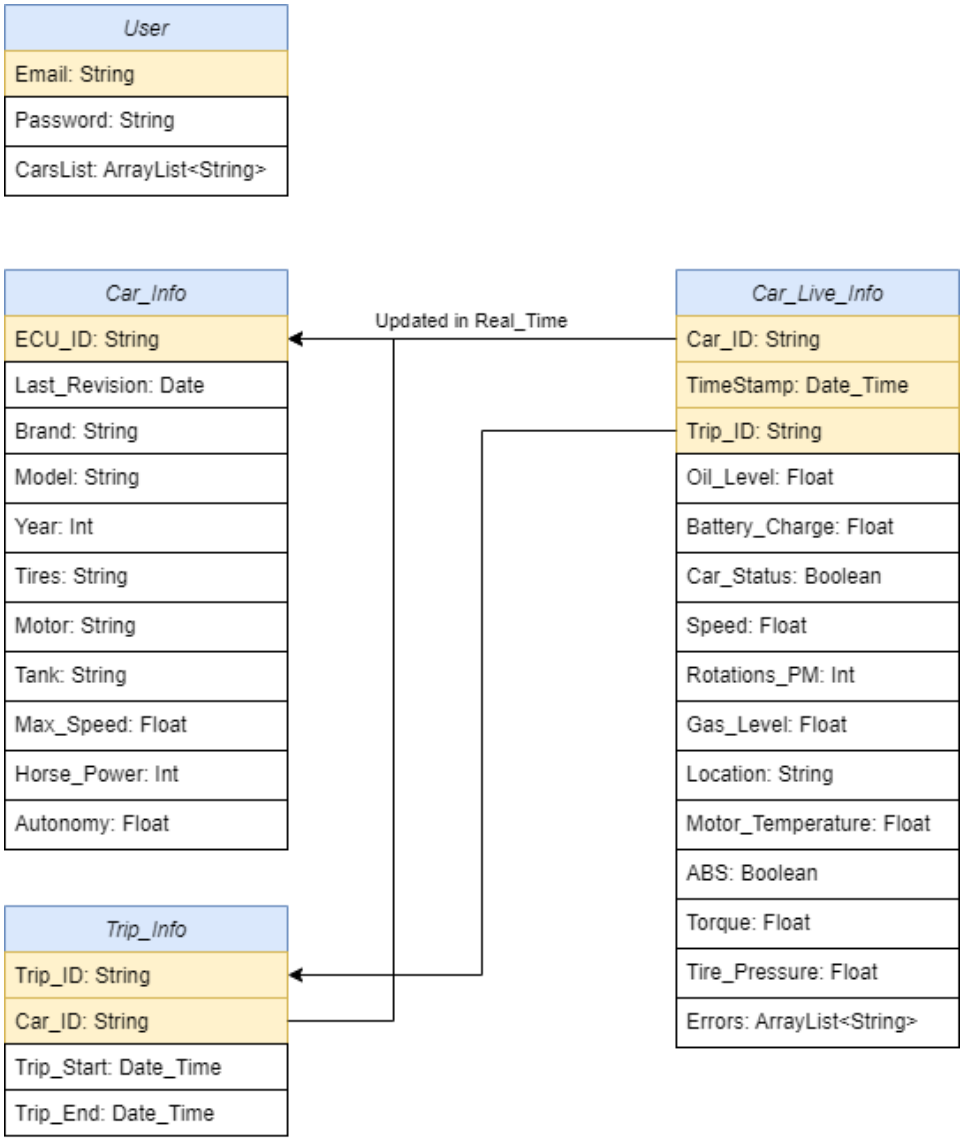

Figure 5. Entity Relationship diagram

5 References and resources

Normal ECU adapter display: https://www.youtube.com/watch?v=Fv4u7o2tP7g