UML Project : Car Management

By: Louis H., Thomas M. and Jules R.

Table of content:

1. Table Structure Diagram

2. Entity-Relation Diagram

3. Use Case Diagrams

4. Gantt Diagram

5. Sequence Diagrams

6. Wireframe Diagrams

7. Activity diagrams

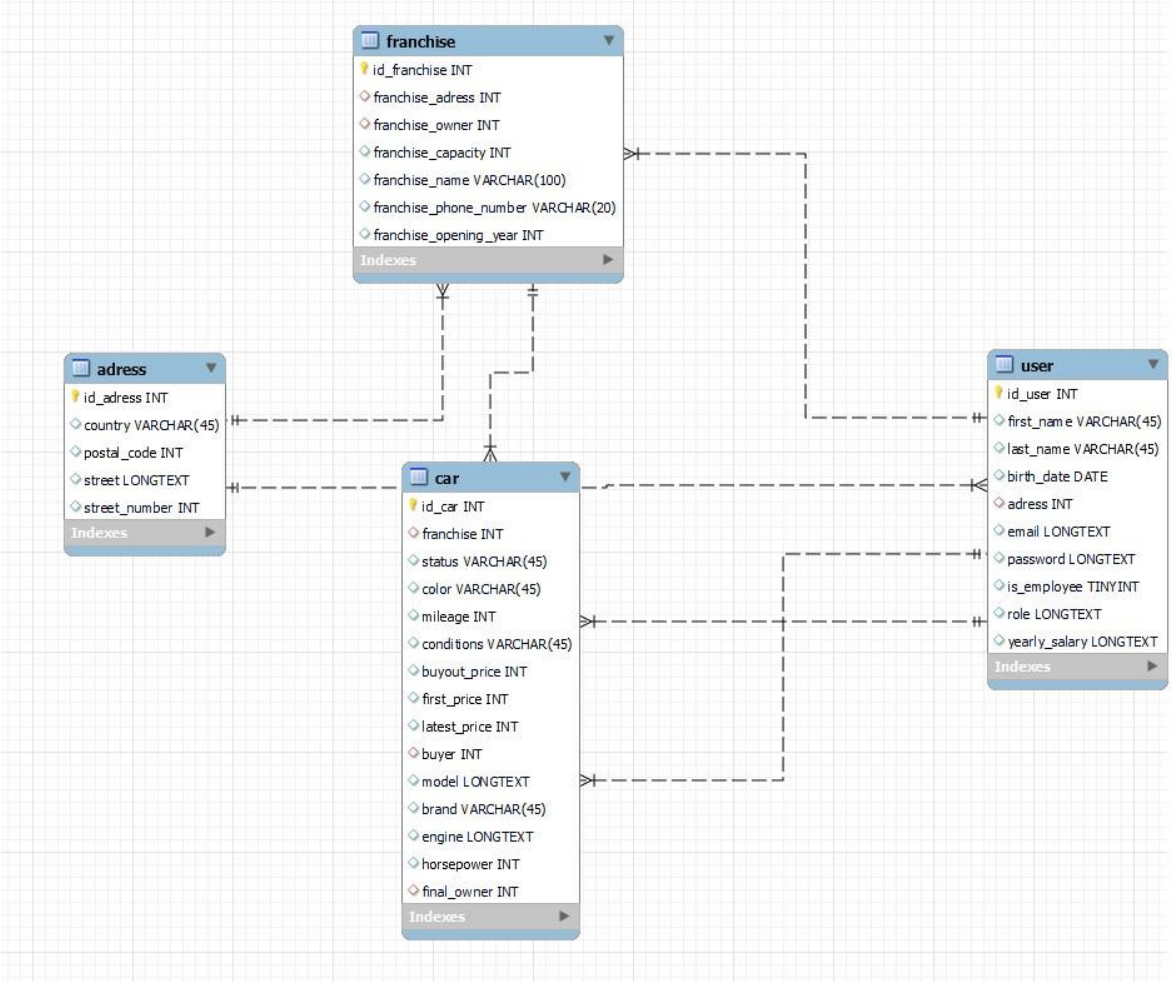
8. Component Diagrams

9. Class diagrams

1. Table structure diagram:

The purpose of this diagram is to visually represent a database

We chose to work with four tables as it allowed us to create a functioning website model with a few key functions without creating too much confusion in our future works.



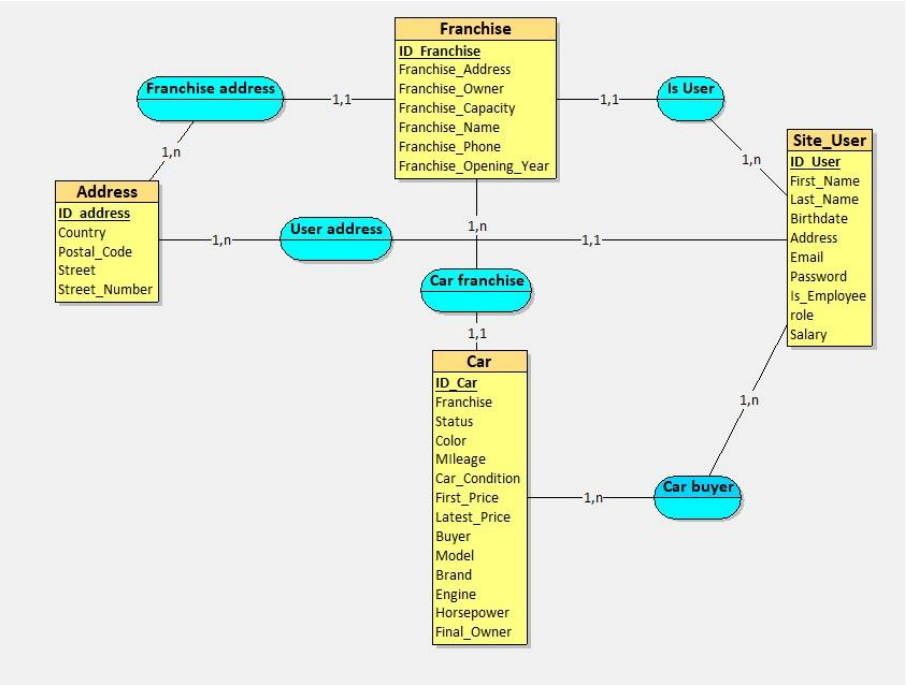
Our four tables are linked using unique IDs created when populating the database. We chose to only use id to link as it is a sure way to avoid potential identification error (for example if we used first and last name for user links, we could have two people named John Doe).

The tables are all linked to each other with the exception of the Address-Car link that we deemed unnecessary for our usage.

2. Entity-Relation Diagram

The purpose of this diagram is to represent the relationships between entities in the database, showing how data is organized and interacts within the system.

- **Franchise**: Represents a car dealership or rental location, containing details like address, owner, capacity, and contact information. It connects to **Address** to specify its location and **Car** to track the vehicles available at each franchise.
- **Address**: Stores information such as the country, postal code, and street details. It connects with both **Franchise** to locate the franchise and **User** to store residential information for users.
- **User**: Represents individual users, with attributes like name, email, and salary. Users are linked to **Address** to indicate where they live and to **Car** to track their purchases.
- **Car**: Represents vehicles, storing details like status, color, mileage, and specifications. Cars are connected to **Franchise** to indicate availability at specific locations and to **User** to track ownership.



This ERD design enables a clear structure for understanding relationships:

1. **Franchise-Address** links franchises to their physical locations.
2. **User-Address** shows users can share or have unique addresses.
3. **Car-Franchise** connects cars to the franchises that manage them.
4. **Car-User** tracks which users own or purchase specific cars.

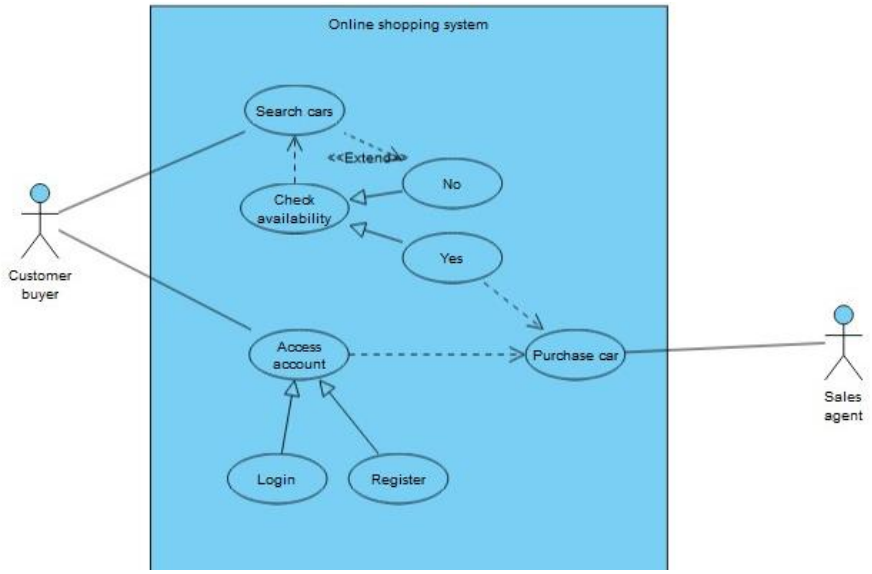
By focusing on simplicity and logical links, the ERD avoids unnecessary complexity while ensuring the database is scalable and functional for future expansions.

3. Use Case Diagrams

These diagrams were created to show how different human actors might interact with our project

a. Shopping for a car

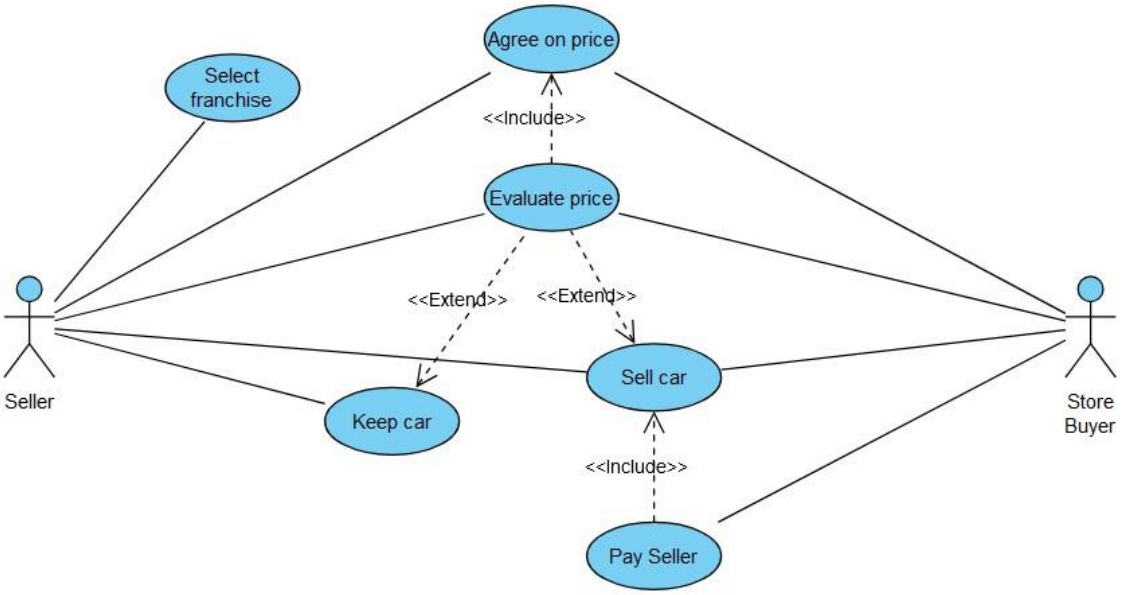
This diagram shows how a user trying to buy a car will interact with our project.



We can see that this action is more complicated and connected than we might have originally thought. For instance, we need constant availability checks on the cars in our Car database as well as a login system to ensure that our buyers have a proper account in our User database.

b. Selling your car

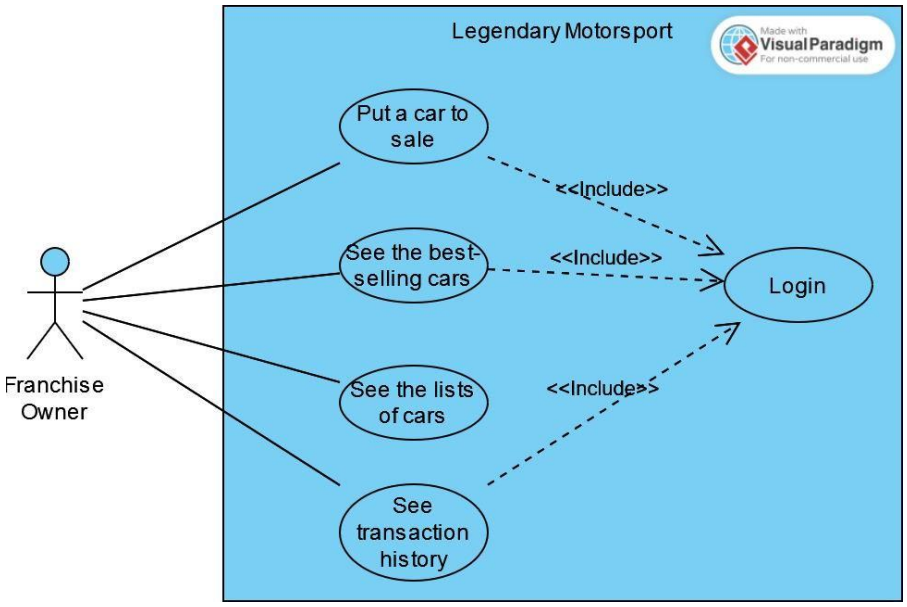
This diagram shows how a user could try to sell his car to a franchise.



We can see that this action requires the intricate co-working of our selling user and our Store employee buying the car from a particular. We can also see that both user have very similar access and that the difference between the roles only happens at the very end when it comes to selling or keeping the car.

c. Owner managament

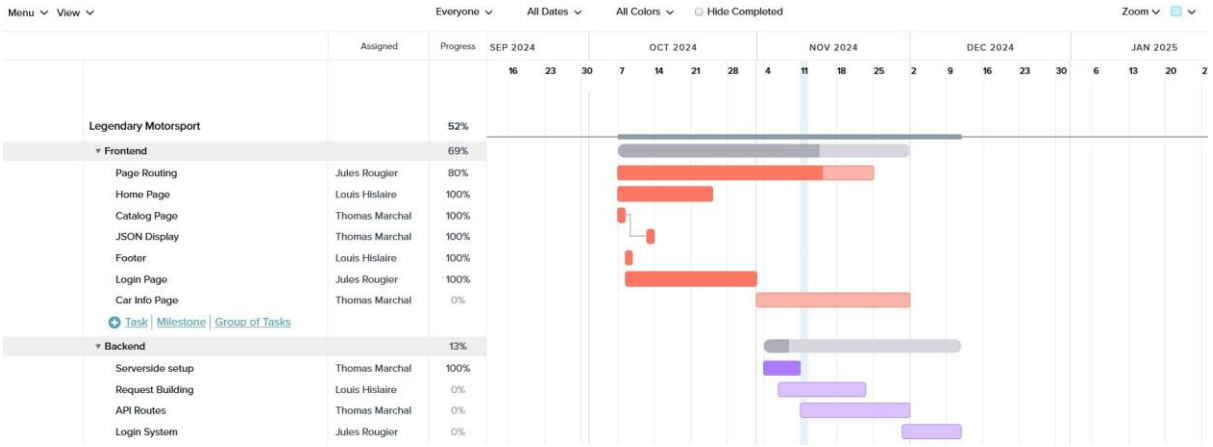
This diagram shows all the possible action a franchise owner can take.



This exemplifies our user system, as we can see that the owner has the same actions as a normal user and can see the list of cars, but we can also see that there are some differences. For instance, the owner also has the “employee” actions such as putting a car to sale but also actions maybe only possible by him such as seeing the complete transaction history of the franchise.

4. Gantt Diagram

This diagram allows to see the work of each of our project programmers



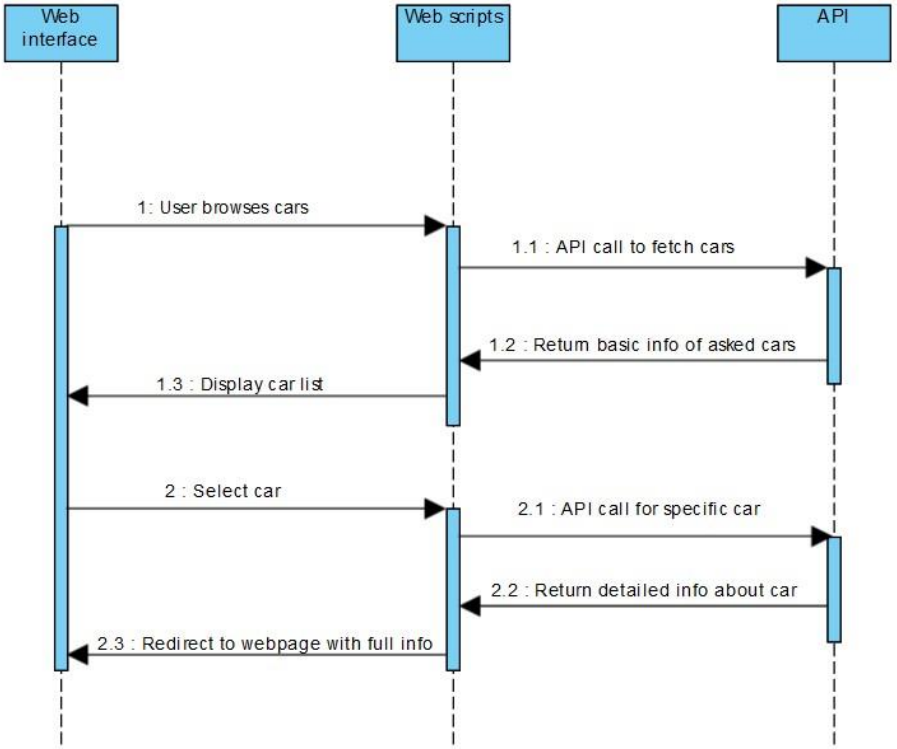
This is extremely useful to see the work of everyone but also the time progression and time estimates for the project. It allows us to have a better understanding of how long this project is and give answers to potential clients.

5. Sequence Diagram :

These diagrams represent the inner working of the project while focusing on the connections between the different layers of the webpage.

A. Browsing cars

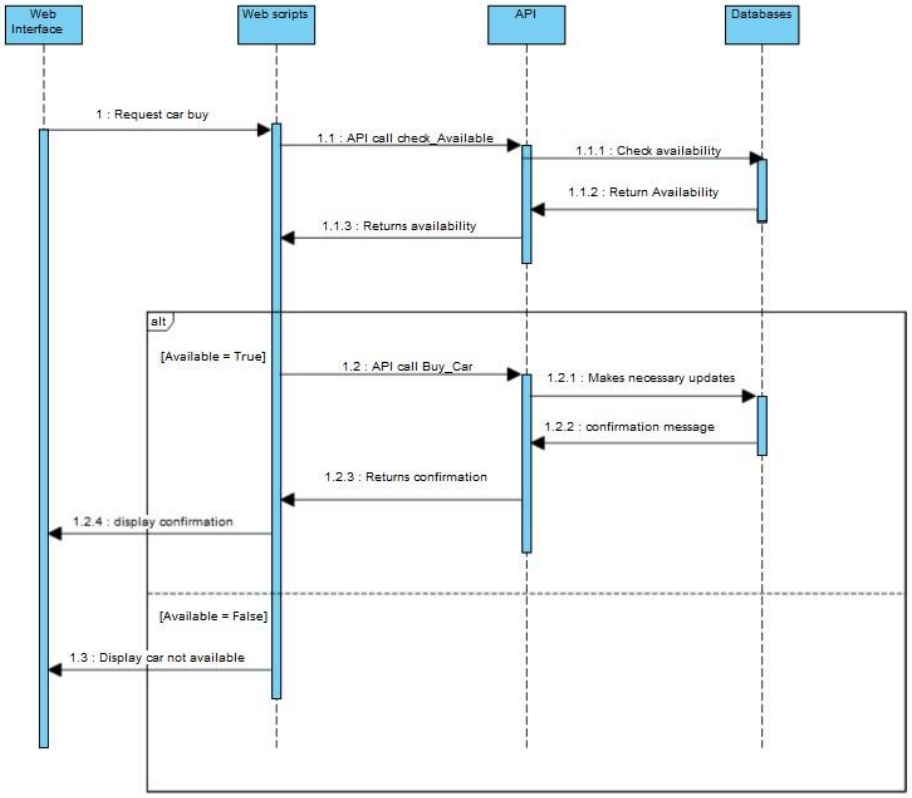
This diagram shows how we go from the web interface to the API in order to display an always updated car list for our users.



We can see for instance that we have two different types of API calls, one for the catalog page fetching all the cars in our database and one for getting additional information on a specific car when the user selects it.

B. Buying car

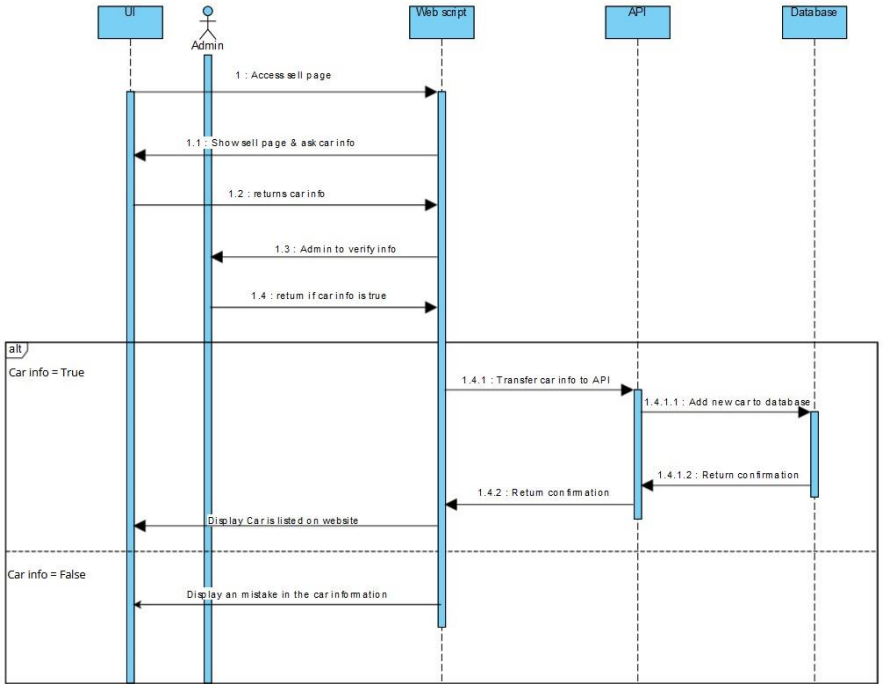
This diagram shows the inner working that goes on when the user chooses to buy a car.



This has a lot of back and forth as there is a need to check for availability right before buying a car then there is another API call to update the database when the car is bought as we cannot sell that car to another customer.

C. Selling your car

This diagram shows the opposite action: selling your car.



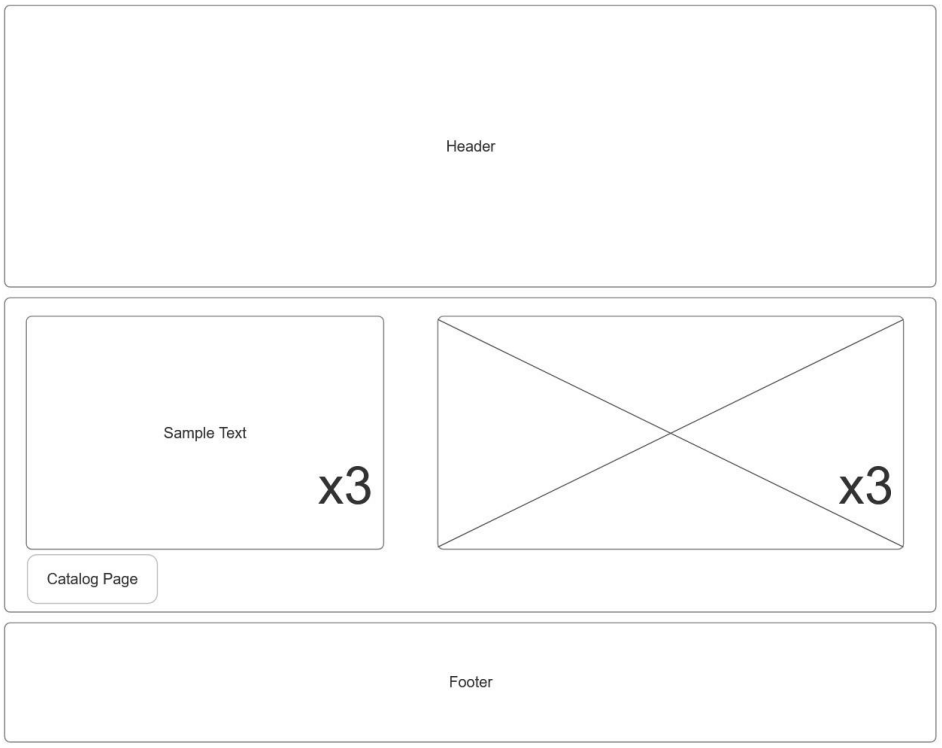
This diagram requires another actor, the administrator, that will ensure that our user is not lying about his car to sell it at a higher price.

6. WIREFRAME DIAGRAMS

These diagrams show a rough draft of our webpages.

A. Home page

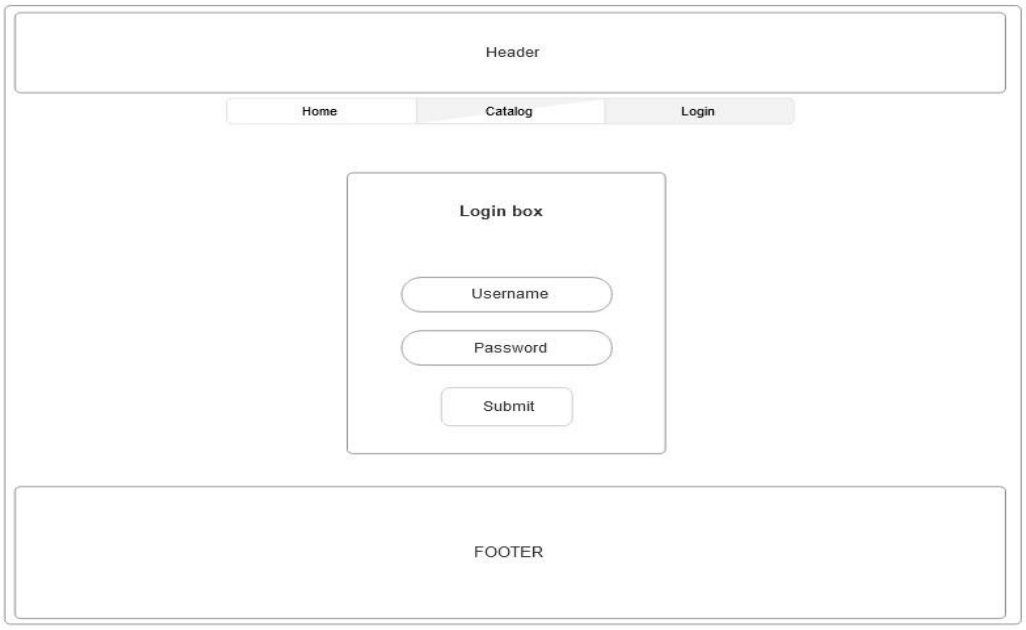
This is our home page, more of a presentation, there are not a lot of interactive elements as it is not useful for that page



This page is more to entice the potential buyers to click on the catalog button and see their available options for car.

B. Login Page

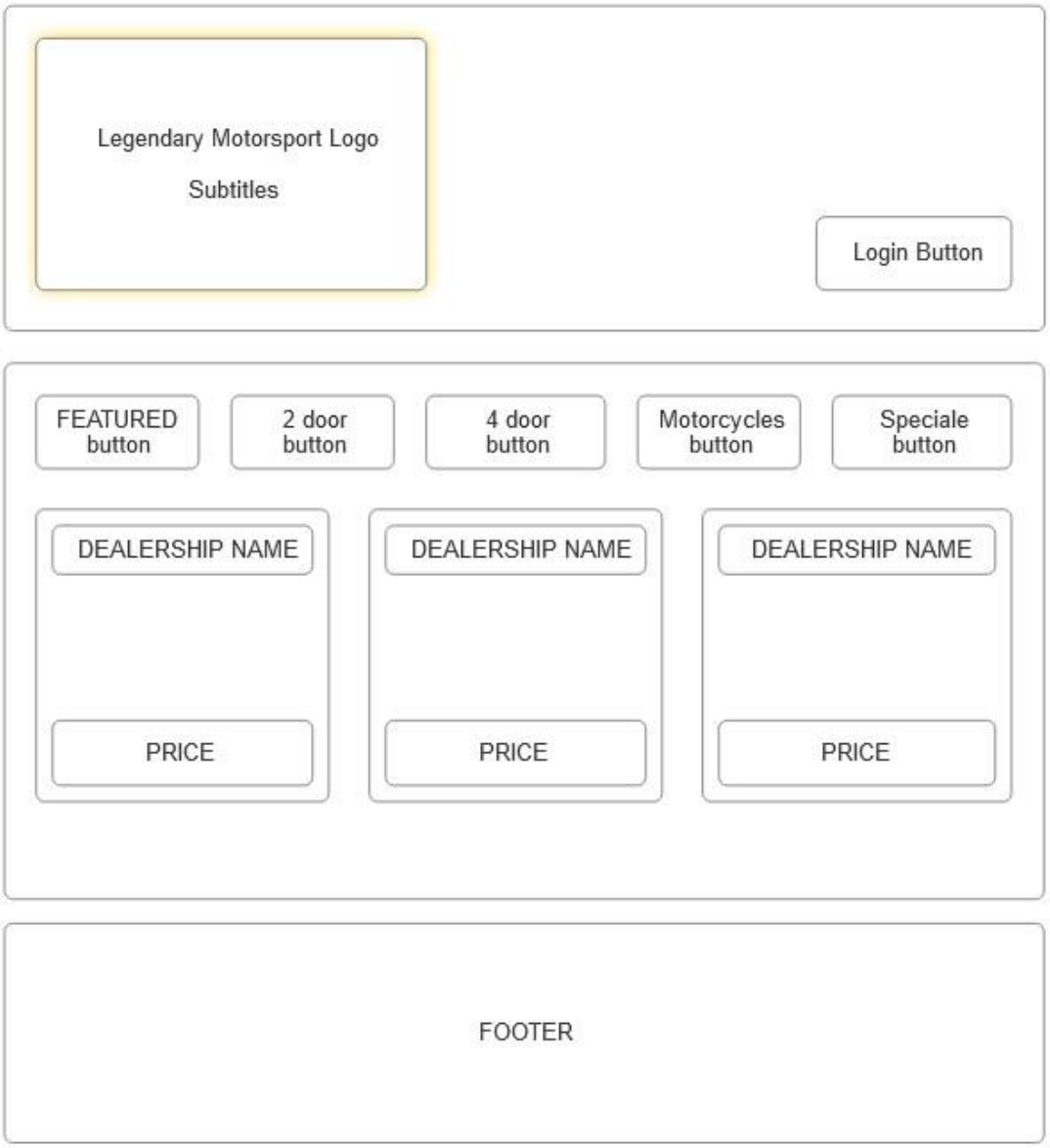
Standard login page, with redirection buttons and a login interface



We kept this page as simple as possible as you want the user to have an experience that is as easy as possible, and the point of this page is not to sell or entice him.

C. Catalog page

This is our most important page, the catalog page, and it contains the most of our interactive elements.



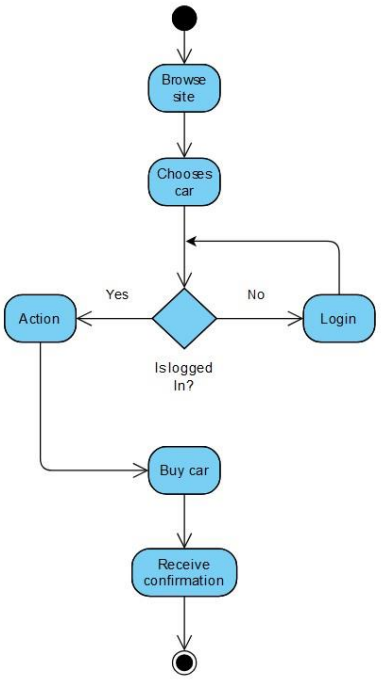
This is made to be as impressive as possible and as intuitive as possible. We have the complete car catalog, with images, franchise locations and prices. To find more information about a certain car you will have to click anywhere on that car box and it will redirect you to another page with additional information. You also have a sorting function to view only the cars that interest you and remove all the other ones.

7. Activity Diagrams

An activity diagram is a flowchart-like representation used in software development modeling to visualize the flow of processes within a system

A. Buying a car

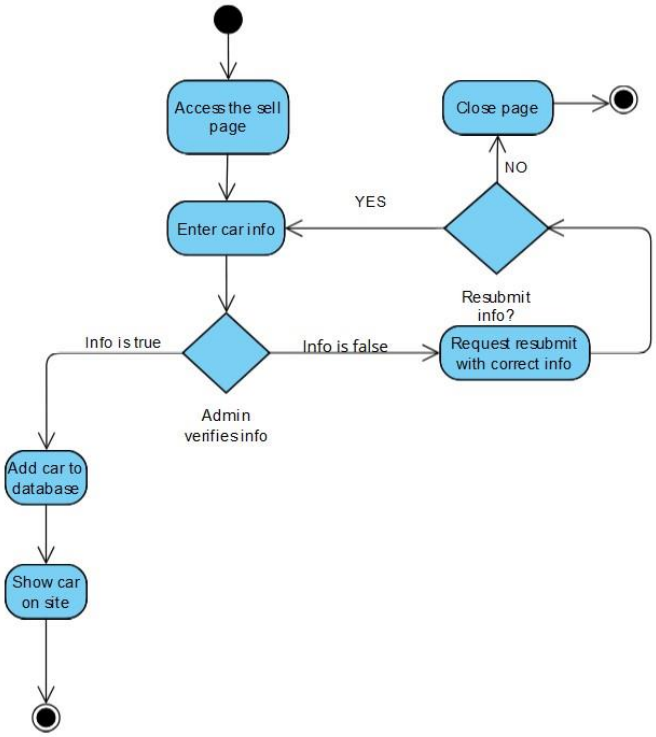
This is a fairly easy diagram showing the way to buy a car from the website.



We have a login check that ensure the user is logged in as it is necessary for our database links.

B. Selling your car

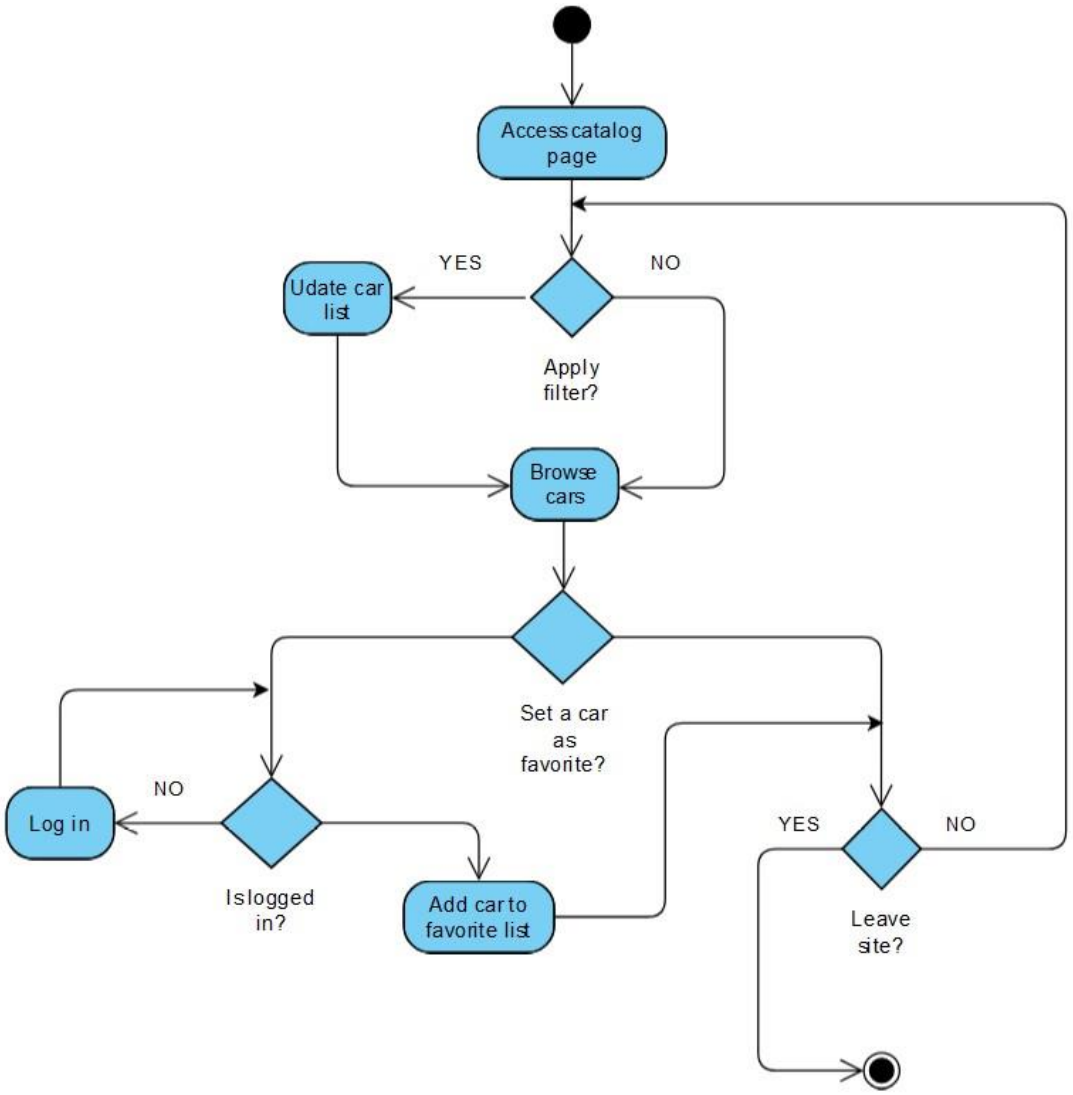
This is a standard diagram that shows how a user might go about selling his car on our website and how it links to our database



We can see that the diagram includes an admin check that verifies the veracity of the user-entered info as allowing customers to have full access to the car selling system would be a massive security breach.

C. Browsing Catalog

This is the diagram that represents a user browsing the car catalog.



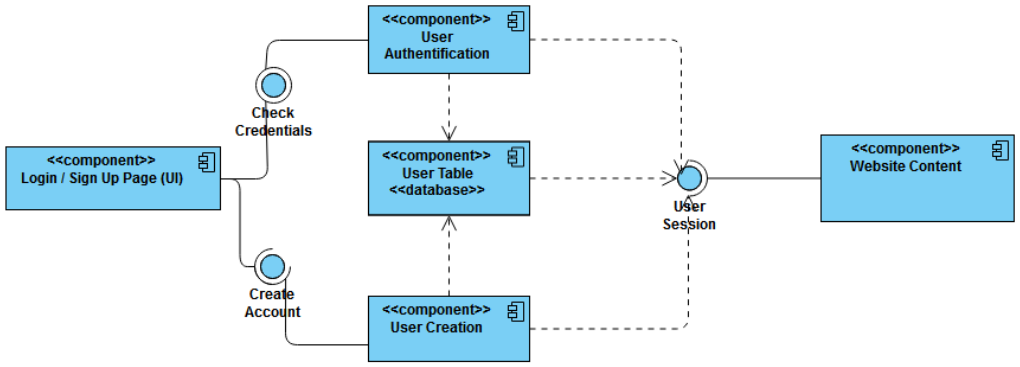
In this diagram we show two important functionalities such as applying a filter but also marking a car as a favorite which requires you to be logged in as it is the only way for use as of now to store the favorite list.

8. Component Diagrams

These diagrams breaks down the actual system under development into various high levels of functionality. Each component is responsible for one clear aim within the entire system

A. Login component

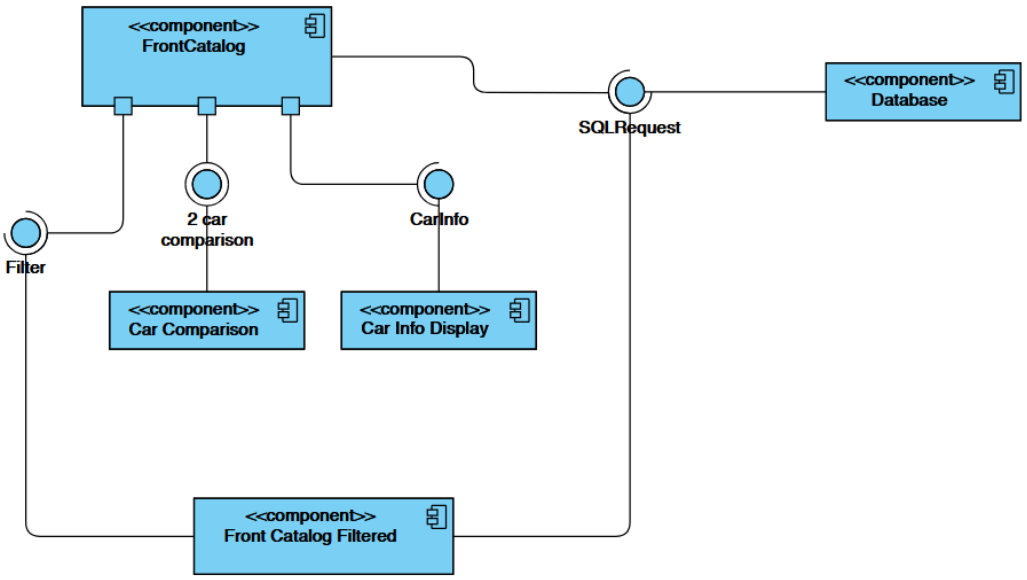
This diagram show the interaction between the login/register component and other components.



The login component interacts with an authentication component while the register component interacts with a user controller. We can also see that that interaction ends with the access to the limited website content.

B. Catalog component

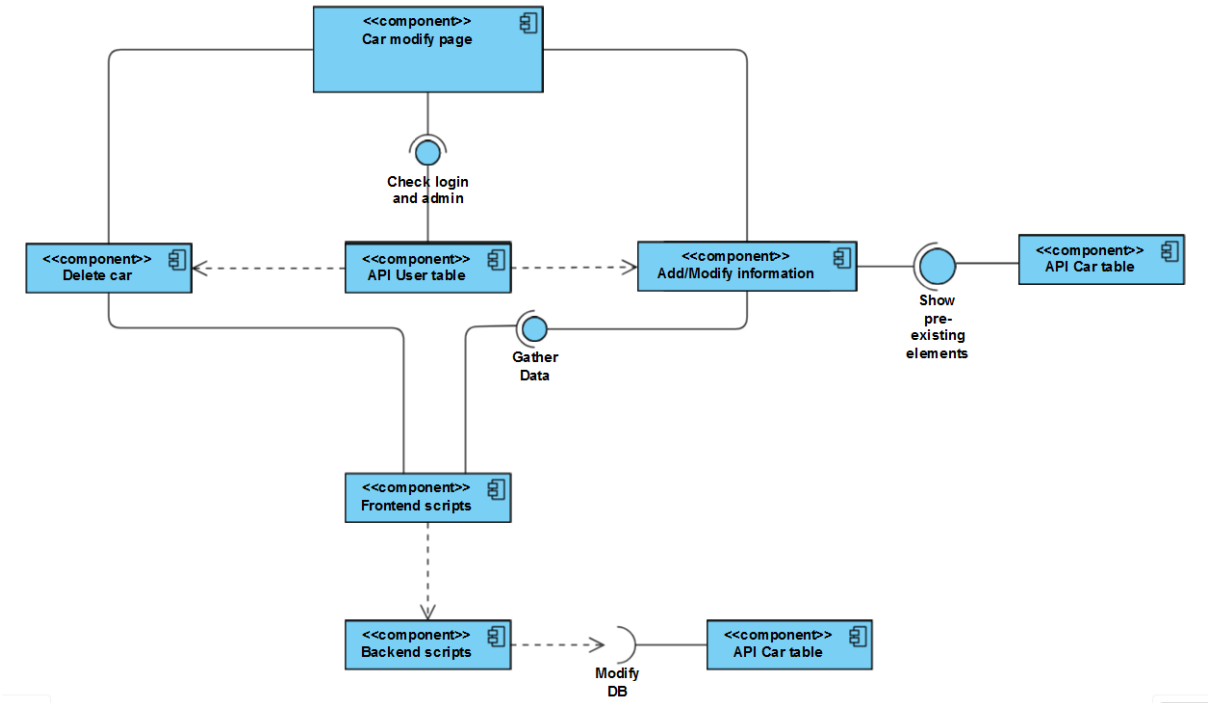
This diagram show the interaction between the Catalog component and other components



We can see an interaction with the database that gets updated whenever a filter is applied and a CarInfo display that redirects the user to a page with information about a specific page.

C. Modify component

This diagram show the interaction between the Catalog component and other components



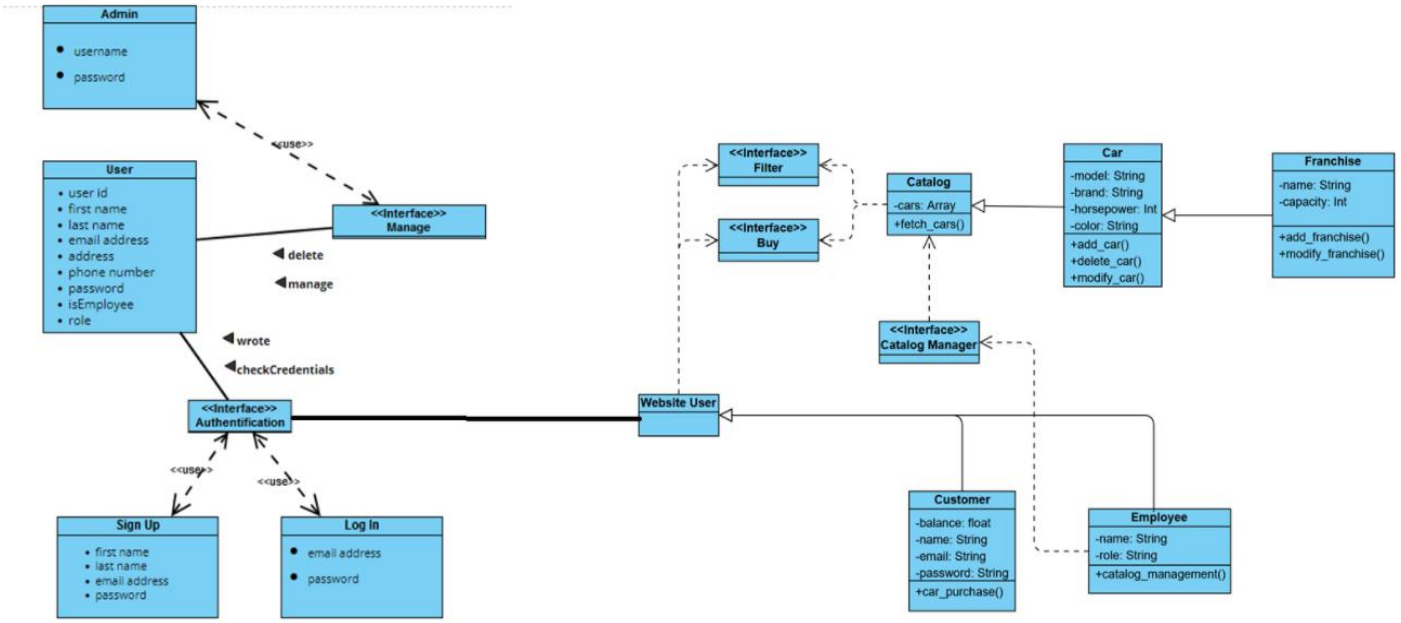
We can see that there are two modification option, the modification of car information or the deletion of the car in our database. Both eventually interact with the front end scripts that apply the wanted changes to the API database.

9. Class Diagrams

These diagrams show the realtion between different classes or object in our project.

A. Authentication Class Diagram

This diagram show the interaction of various classes within our project.



As you can see, the main feature of our project is the catalog and everythin builds around it. This is why we link it to the login part of the diagram and it also gets connected to our car controller that allows chosen users to modify the car database.