These are some details of the project to know before starting:

The project has 3 users:

Feel free to login with any of these credentials

1. Username: user1
   Password: password1

2. Username: user2
   Password: password2

3. Username: user3
   Password: password3

User 3 has no information about cars, feel free to add any to it

Authentication:

- JWT tokens are used to authenticate users in the system.
- A Token is generated with a secret key.
- The token can be found in sessionStorage, it stores:
    - Token
    - userId
    - Username
- Once a user logs out, the token is unavailable
- Routes are protected. Without logging in, you can't access the database.
- The Auth Service is in services repository

Features:

- Listing all cars:
    - All cars are listed initially, you can also refresh the page to list every car, or search empty parameters
- Adding a car:
    - Cars are added by make, model and year, with optional stocklevels.
    - The correct validation is in place to ensure no bad inserts occur.
    - Cannot insert a duplicate car
    - Year can only be between 1886 and the current year. (cannot have a car that has not been released yet)
- Deleting a car:
    - Cars are deleted via the button.
- Update a car:
    - Can update a cars stock levels according to certain rules (no negatives or non integers)
- Search car by make and model:

- Search works via fuzzy search:
    - Case insensitive
    - Works with “starts with” logic : if you search “mer” then any record starting with mer will match

APIs:

All SQL queries are in Data/CarRepository
All APIs are in Controllers/CarsController
All API calls from frontend are in api/api.js

- GET /api/Cars
    - Fetches all cars, and returns a list
- GET /api/Cars/searchById
    - Takes a make, model and year, and returns the car ID
    - Used mainly for deletes and edits
    - Returns 0 if nothing
- POST /api/Cars/
    - Insert car api
- DELETE /api/Cars/{id}
    - Deletes the corresponding car
- PUT /api/Cars/{id}
    - Updates the car
    - Returns rows affected
- GET /api/Cars/search
    - Takes a make and model and searches for it in the database
    - Uses fuzzy search
    - Returns a car if matched

Database:

- The database schema at CarStocks.db is as follows:

Cars Table

Id: unique id acting as the primary key for each car. Integer
Make: the make of the car as a string
Model: the model of the car as a string
Year: the year of the car as an integer
UserId: Id of the manager associated with this car (at the dealership)

User Table

Id: UserID having to do with a manager
Username: username of user
Password: password of user

Future Improvements:

- Confirm delete popup
- Instead of UserID, use the dealership ID, so that many users part of the same dealership can access the same database

- Hash Password
- Sign up as a user for a dealership