Realsoft Car Log System

Welcome to the system specification document for our car and driver management application. This application consists of three main services: car service, driver service, and car log service. Each service performs a specific function in managing our car and driver resources.

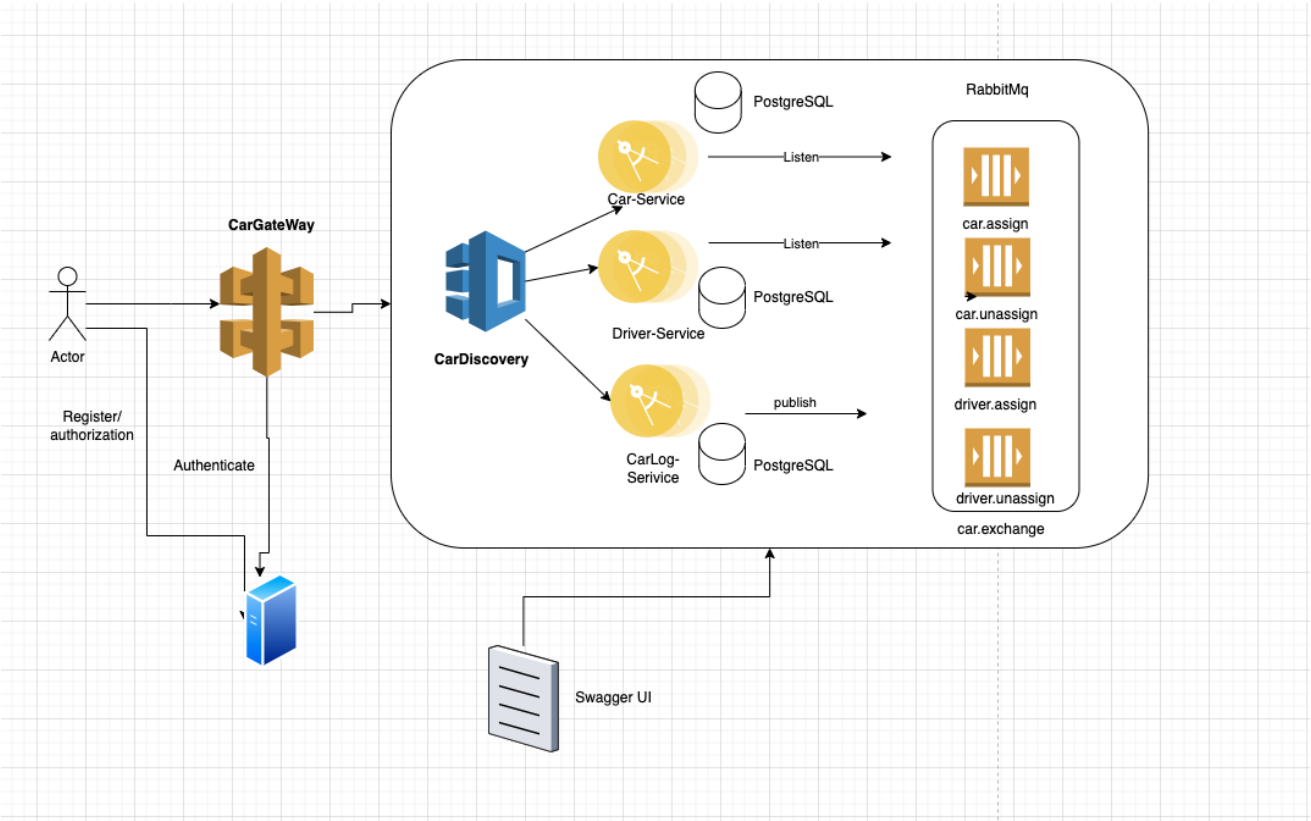
When a user creates a log from the car log service, a message is published to the car.assign and driver.assign queues via RabbitMQ. This message triggers the car and driver services to assign a car and driver to the logged duty. The assigned car and driver information is stored in PostgreSQL for future reference.

Our application also includes an Auth server where users can register and manage their account information. This server ensures that only authorized users have access to our application resources.

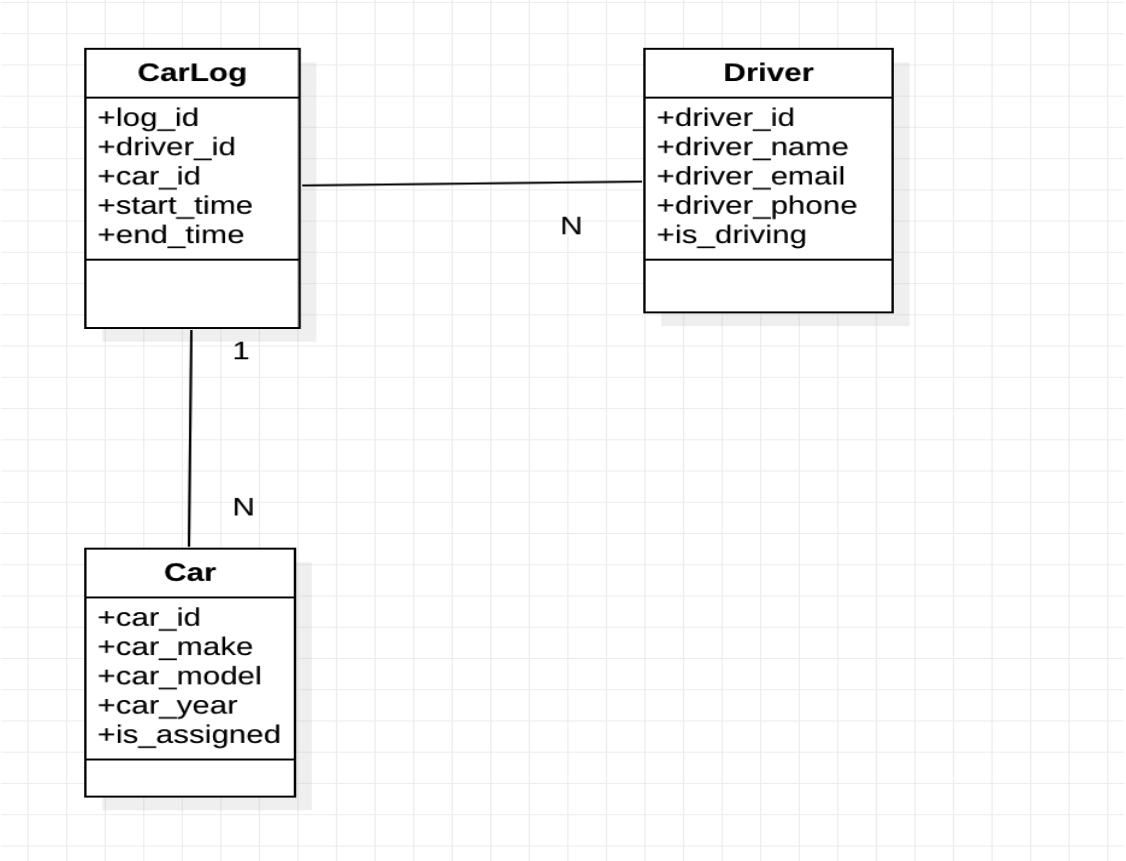
To help you better understand the system design and how to use our application, we have included the ER diagram, sequence diagrams, and system design picture. These diagrams provide a visual representation of our application's data flow and architecture.

In the following sections, we will discuss in detail how to use our application, including how to register for an account, how to create logs, and how to manage cars and drivers. We will also provide technical information about our application's architecture and data flow.

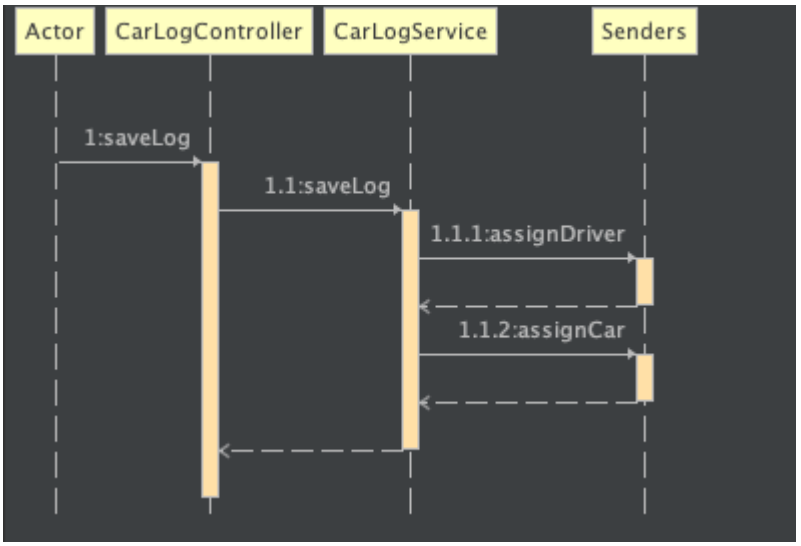
Here for better Understanding I will mention the high level design of my Application.



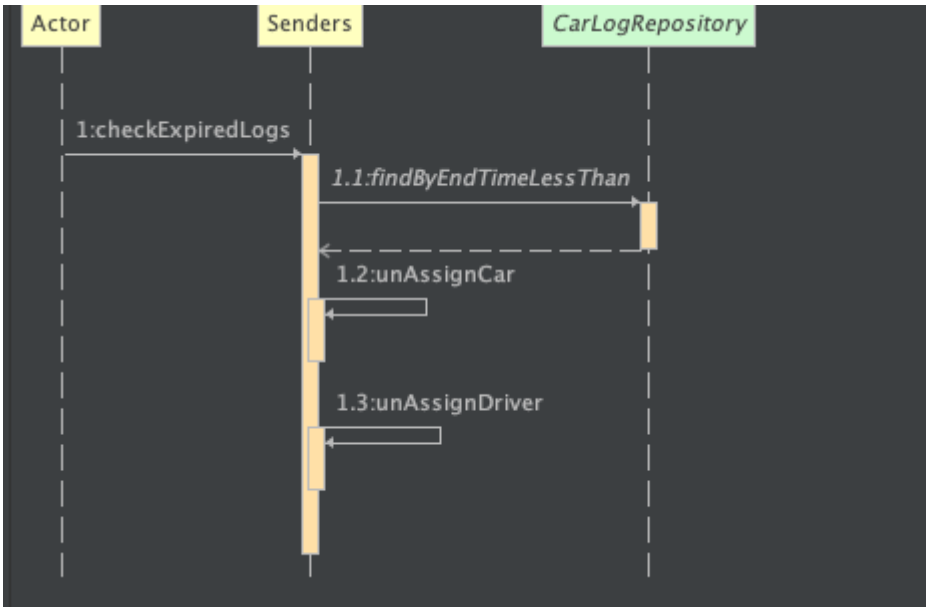
Here I am also attaching ER diagram of our application for better understanding of application



Here I am pasting some sequence diagrams

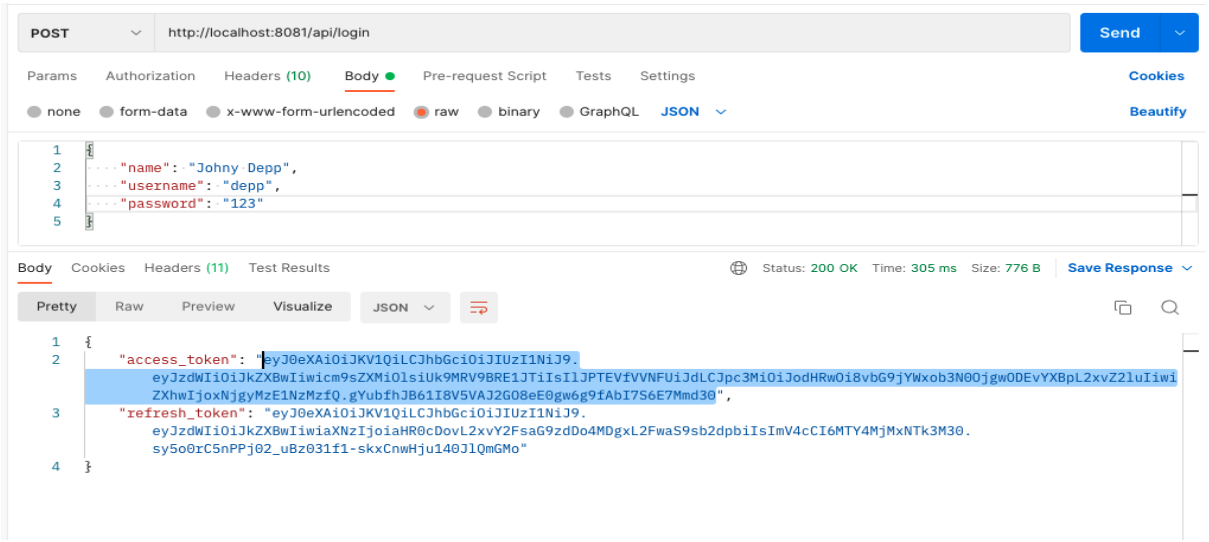


this is to create a log from Carlog service it will push messages to queues



This sequence of Scheduled tasks in RabbitMq checks the database each hour if the time of logs are expired and pulls the expired logs and makes the driver and car for the next logs.

1.0 First of All user will register to the auth server and get the token and then they could actually request to api gateway to perform any task



Above is the way of sending request to login to get tokens

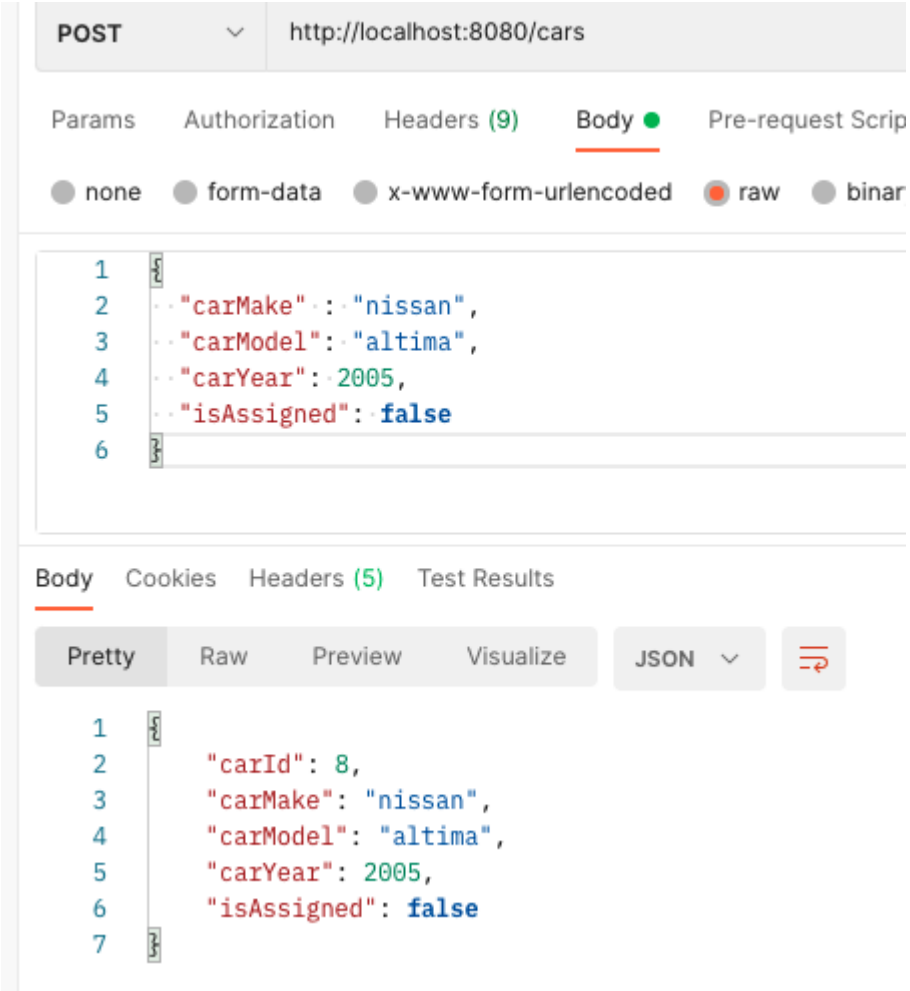
For simplicity I have created some users in the kickstart of the application to make Auth server simple.

```java
@Bean
CommandLineRunner run(UserService userService){
    return args -> {
        userService.saveRole(new Role( id: null,  name: "ROLE_USER"));
        userService.saveRole(new Role( id: null,  name: "ROLE_MANAGER"));
        userService.saveRole(new Role( id: null,  name: "ROLE_ADMIN"));
        userService.saveRole(new Role( id: null,  name: "ROLE_SUPER_ADMIN"));

        userService.saveUser(new User( id: null,  name: "Johny Depp",  username: "depp",  password: "123", new ArrayList<>()));
        userService.saveUser(new User( id: null,  name: "Will Smith",  username: "smith",  password: "123", new ArrayList<>()));
        userService.saveUser(new User( id: null,  name: "Bradley Cooper",  username: "cooper",  password: "123", new ArrayList<>()));
        userService.saveUser(new User( id: null,  name: "Ryan Reynolds",  username: "reynolds",  password: "123", new ArrayList<>()));
        userService.saveUser(new User( id: null,  name: "Tom Hanks",  username: "hanks",  password: "123", new ArrayList<>()));
        userService.saveUser(new User( id: null,  name: "Tom Holland",  username: "holland",  password: "123", new ArrayList<>()));

        userService.addRoleToUser( username: "depp",  roleName: "ROLE_USER");
        userService.addRoleToUser( username: "smith",  roleName: "ROLE_MANAGER");
        userService.addRoleToUser( username: "cooper",  roleName: "ROLE_MANAGER");
        userService.addRoleToUser( username: "reynolds",  roleName: "ROLE_USER");
        userService.addRoleToUser( username: "hanks",  roleName: "ROLE_SUPER_ADMIN");
        userService.addRoleToUser( username: "depp",  roleName: "ROLE_ADMIN");
    };
```

1.1 After that we will send request to our services through the api-gateway



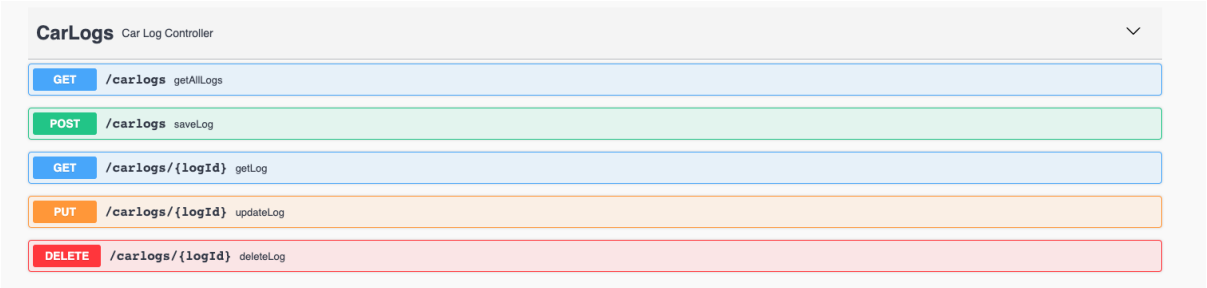
2.0 We are using RabbitMq Here to communicate with our car service and driver service
for that we need for queues they are car.assign, car.unassign, driver.assign, driver.unassign
I have chose to use RabbitMq Ui to create exchange and bind the queue to that exchange with specific binding keys

Overview				Messages			Message rates		
Name	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
car.assign	classic	D	idle	0	0	0			
car.unassign	classic	D	idle	0	0	0			
driver.assign	classic	D	running	0	1	1		36/s	0.00/s
driver.unassign	classic	D	idle	0	0	0			

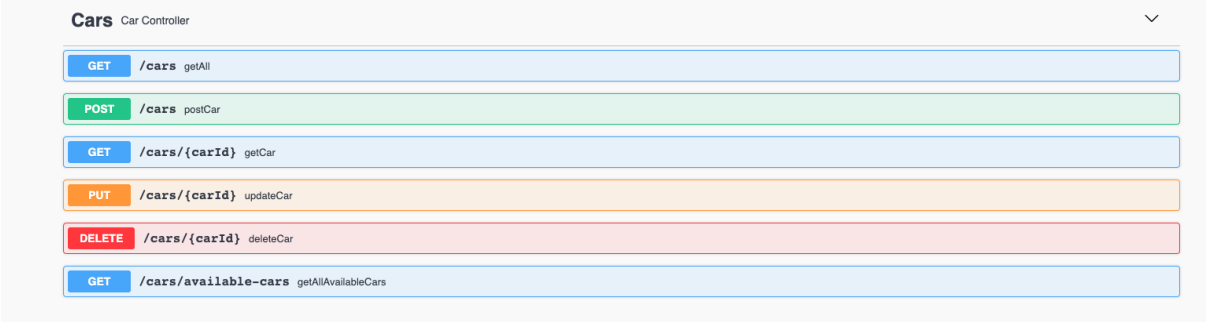
+/-

3.0 Here are the end points that are going to be used in our system

CarLogs Api



Cars Api



Drivers APi