# Car Rental

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Change Details** | **Change By** | **Change Date** |
| 0.1 | Car Rental Demo | Jimmy Luo | 20191221 |

Table of Contents

[1. Assumption 1](#_Toc27919523)

[2. Scope 1](#_Toc27919524)

[3. Solution 1](#_Toc27919525)

[4. Business Work Flow 2](#_Toc27919526)

[5. EDR Diagram 2](#_Toc27919527)

[6. Sequence Diagram 3](#_Toc27919528)

[7. API Design 4](#_Toc27919529)

[8. Source Repository 5](#_Toc27919530)

[9. Test Cases 5](#_Toc27919531)

[10. CI/CD Work Flow 5](#_Toc27919532)

[11. Google Cloud App host 8](#_Toc27919533)

[12. Future Version 10](#_Toc27919534)

## 

## Assumption

1. Assumed we only got 2 models of cars, and each model only has 2 cars.
2. Assumed we only have few customers so we only need to deploy our app to a single machine, and handle the concurrency with jvm technology.

## Scope

1. Implement sign up function, which allows user to sign up
2. Implement sign in function, which allows user to sign in
3. Implement car search function, which allows user search the car by rent date and return date only
4. Implement Place order function, which will generate the order to user after user choose a car
5. I have used the azure trial quota before, so I don’t have the trial account now, I am going to use google cloud.

## Solution

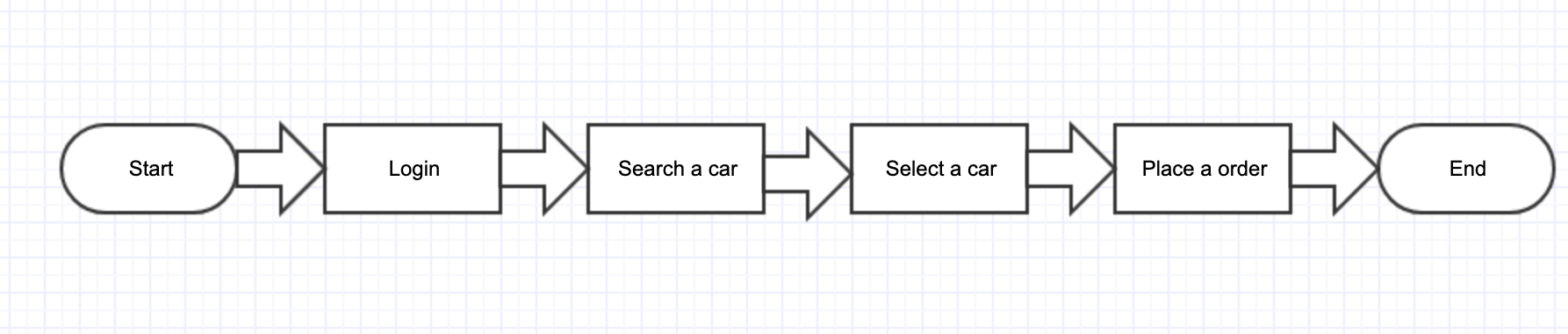
View: Theameleaf

Web Service: Spring boot + Spring MVC

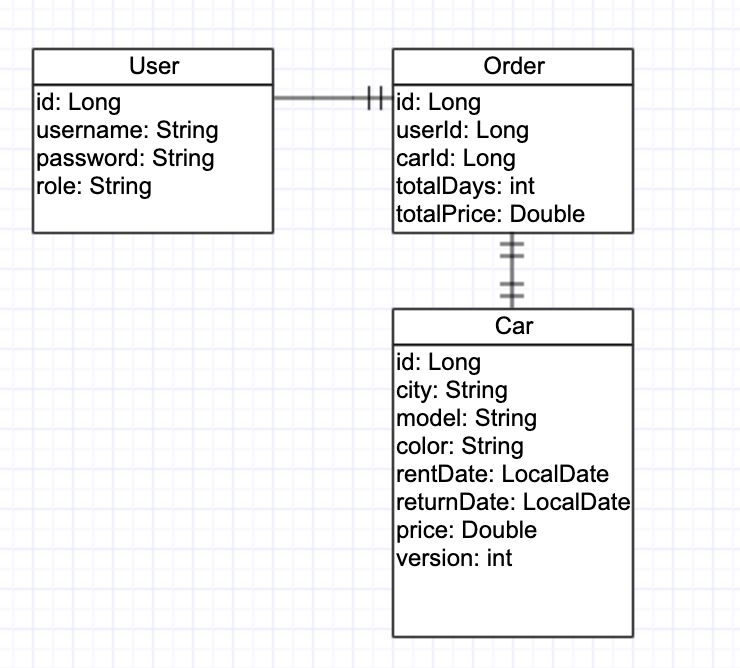
DB: H2

Please note that using the above technologies coupling frontend and backend in the single machine is just for demo purpose

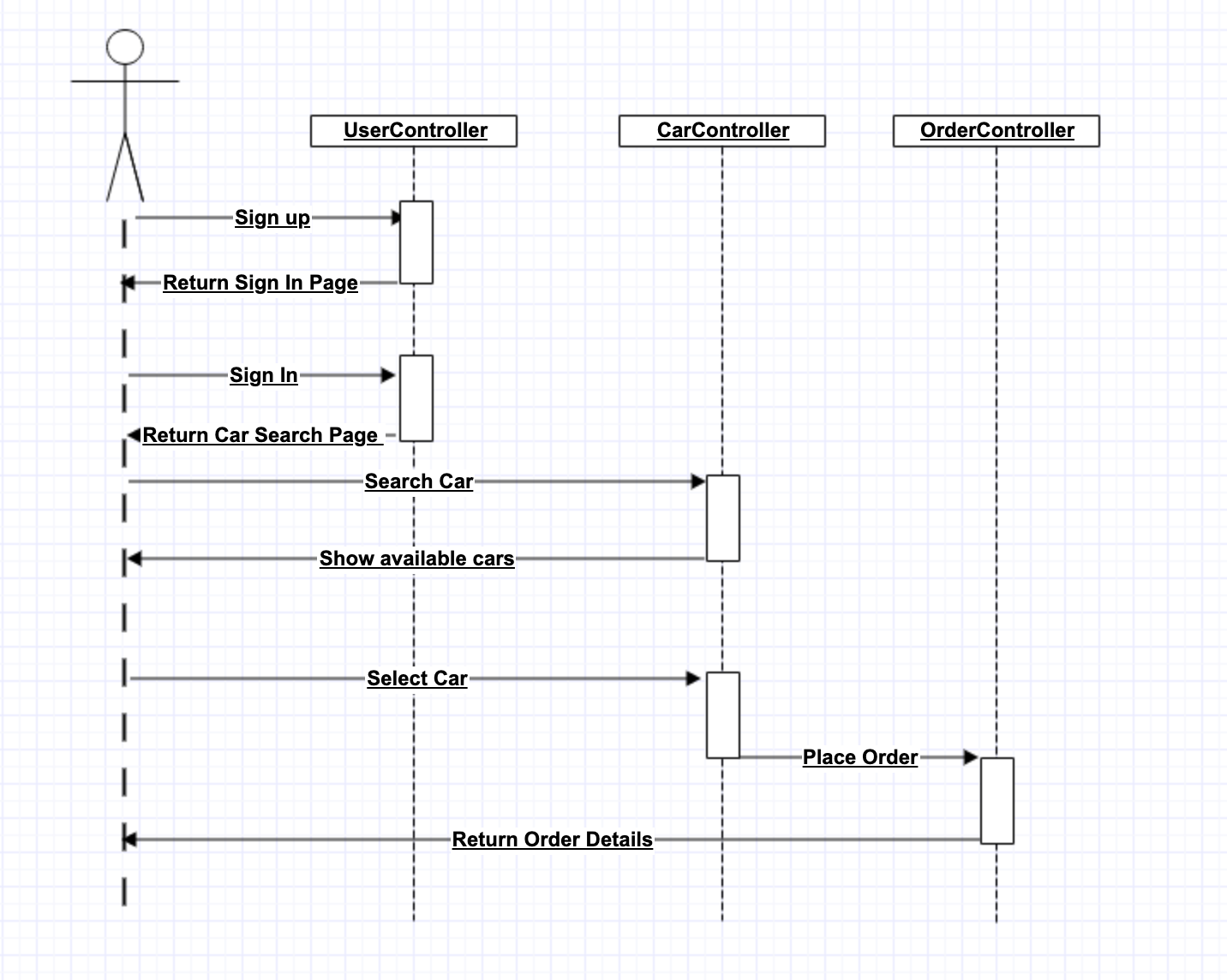
## Business Work Flow



## EDR Diagram



## Sequence Diagram



## API Design

|  |  |
| --- | --- |
| path: | /saveuser |
| request body: | {  "signupForm":{  "username": "jimmyluo",  "password": "abcd1234",  "passwordCheck": "abcd1234"  } } |
| response: | Success: login  Failed: prompt the error message |

|  |  |
| --- | --- |
| path: | /search |
| request body: | {  "searchRequest":{  "rentDate": "2019-12-20",  "returnDate": "2019-12-23"  } } |
| response boy: | {  "cars":  {"car":[  {"carId": "1001",  "city":"Guangzhou",  "color":"Red",  "carModel":"Toyota Camry",  "rentDate":"2019-12-15",  "returnDate":"2019-12-19",  "price":"100.00"},  {"carId": "1002",  "city":"Shenzhen",  "color":"Red",  "carModel":"BMW 650",  "rentDate":"2019-12-10",  "returnDate":"2019-12-15",  "price":"100.00"}  ]  } } |

|  |  |
| --- | --- |
| path: | /order/{carId} |
| request body: | {  "searchRequest":{  "rentDate": "2019-12-20",  "returnDate": "2019-12-23"  } } |
| response boy: | {"order":  {"id": 1,  "city":"Guangzhou",  "carModel":"Toyota Camry",  "carColor":"Red",  "rentDate":"2019-12-15",  "returnDate":"2019-12-19",  “totalDays”: 4,  "unitPrice":100.00,  “totalPrice”:400  “ownedBy”:”jimmyluo”}  } |
|  |  |

## Source Repository

<https://github.com/hydroblok/carrental_demo>

## Test Cases

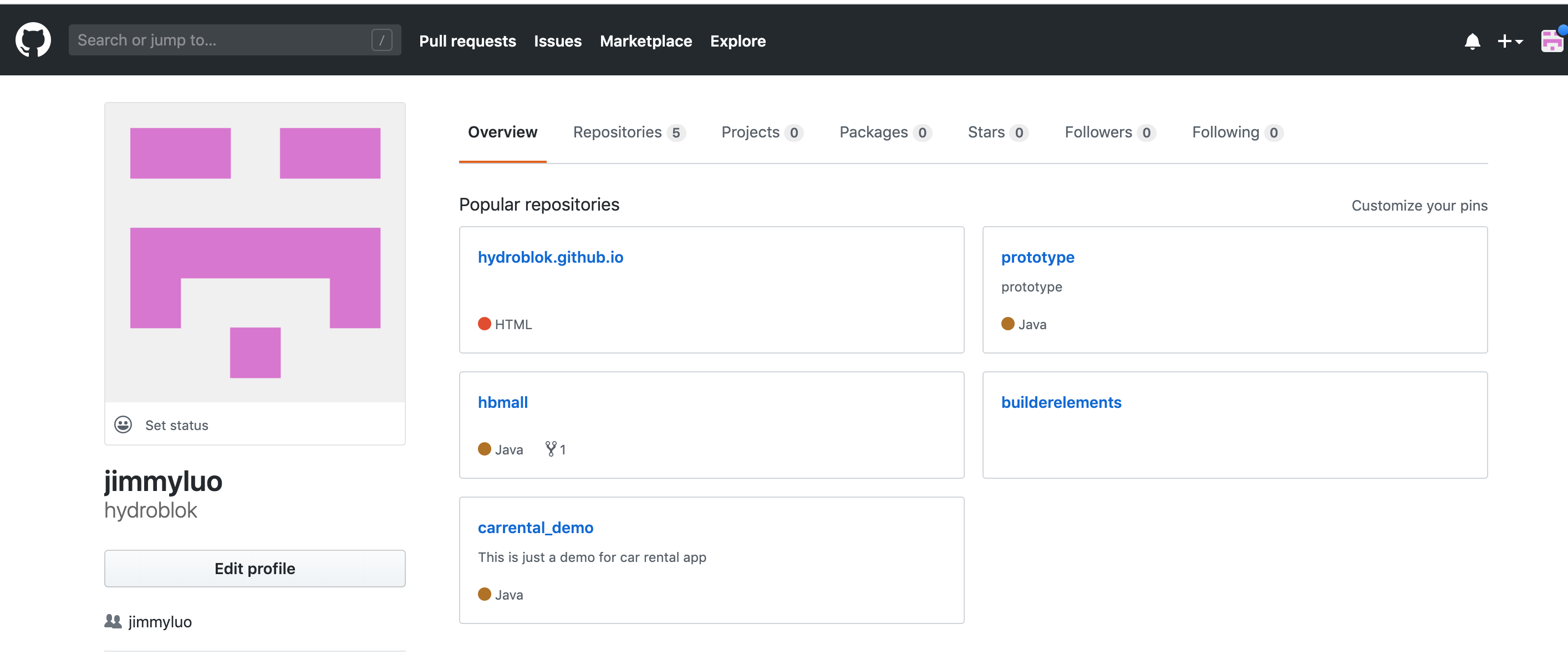
Please refer the details from the following link

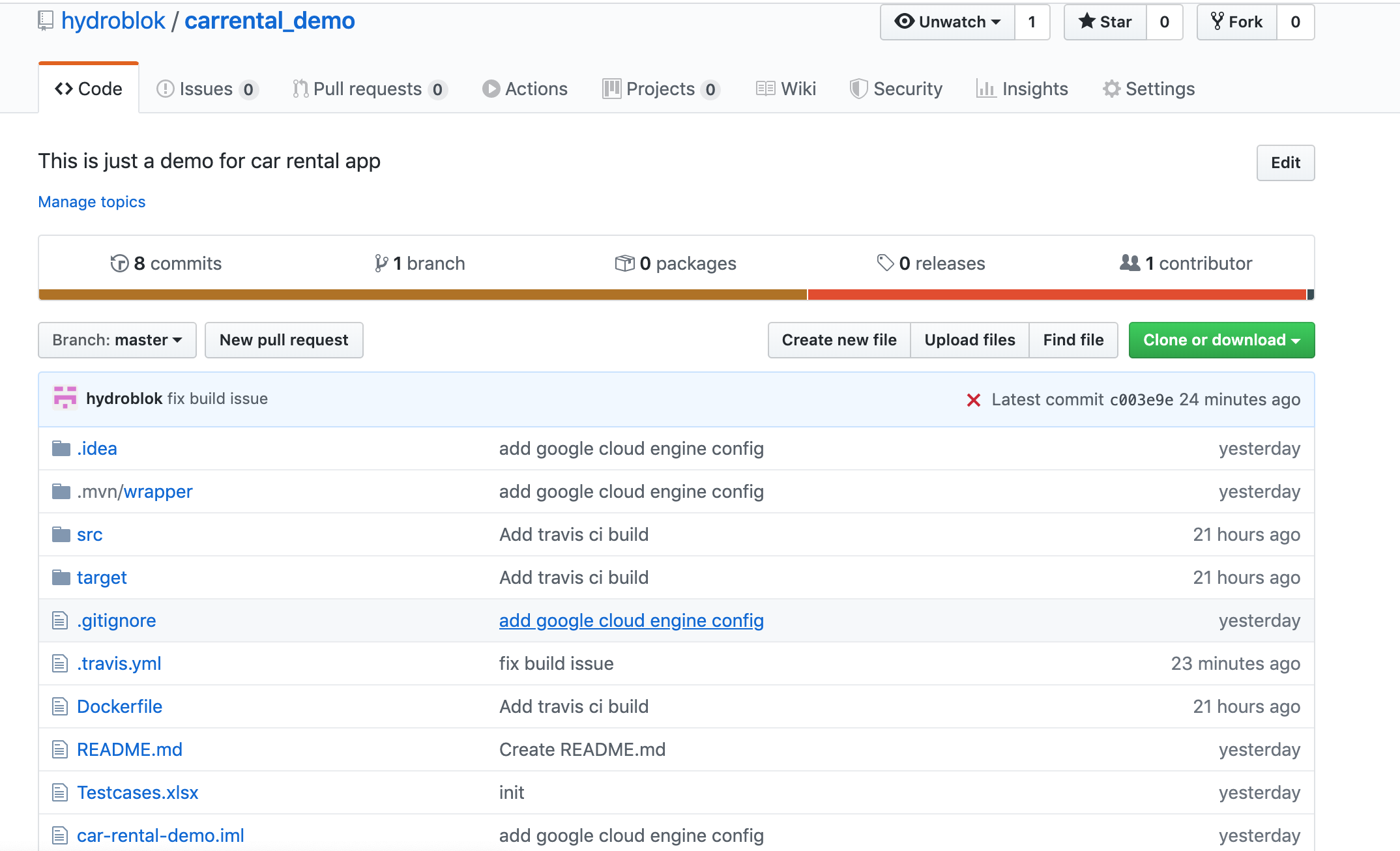
<https://github.com/hydroblok/carrental_demo/tree/master/TestCases.xlsx>

## CI/CD Work Flow

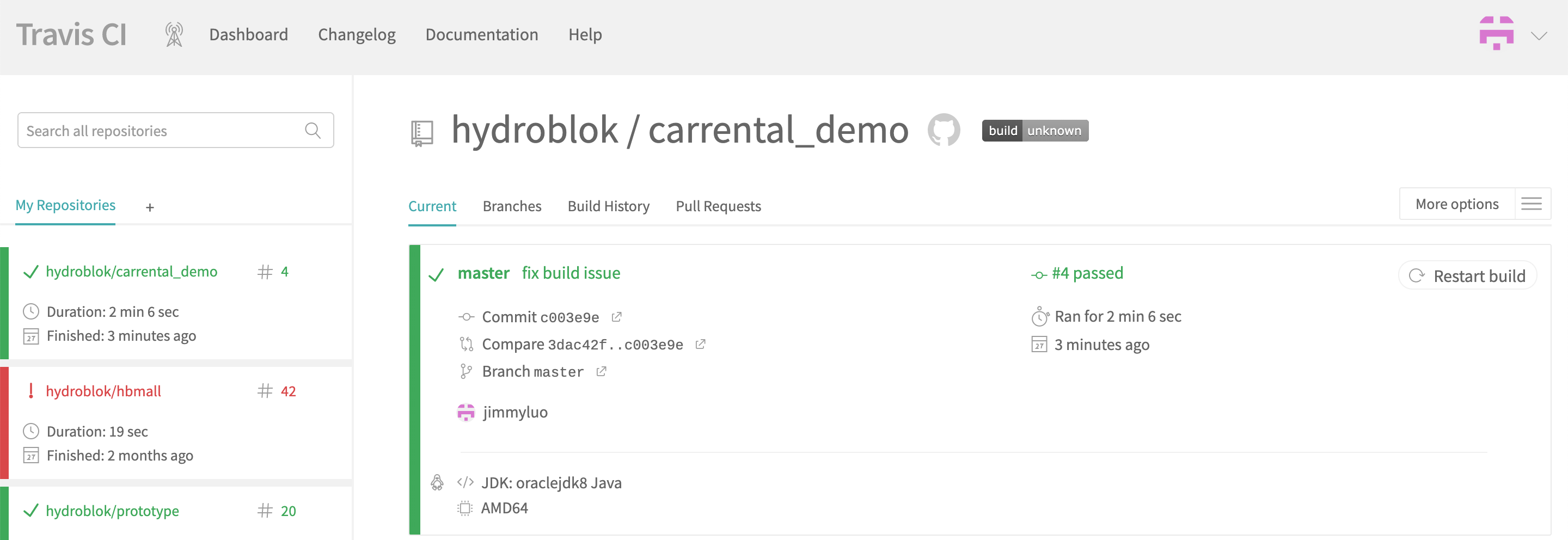
Push code to Github -> Travis run the build and test -> Do code analysis in Sonnar Cloud -> Push the image to Docker Hub -> Deploy to Google Cloud

Github

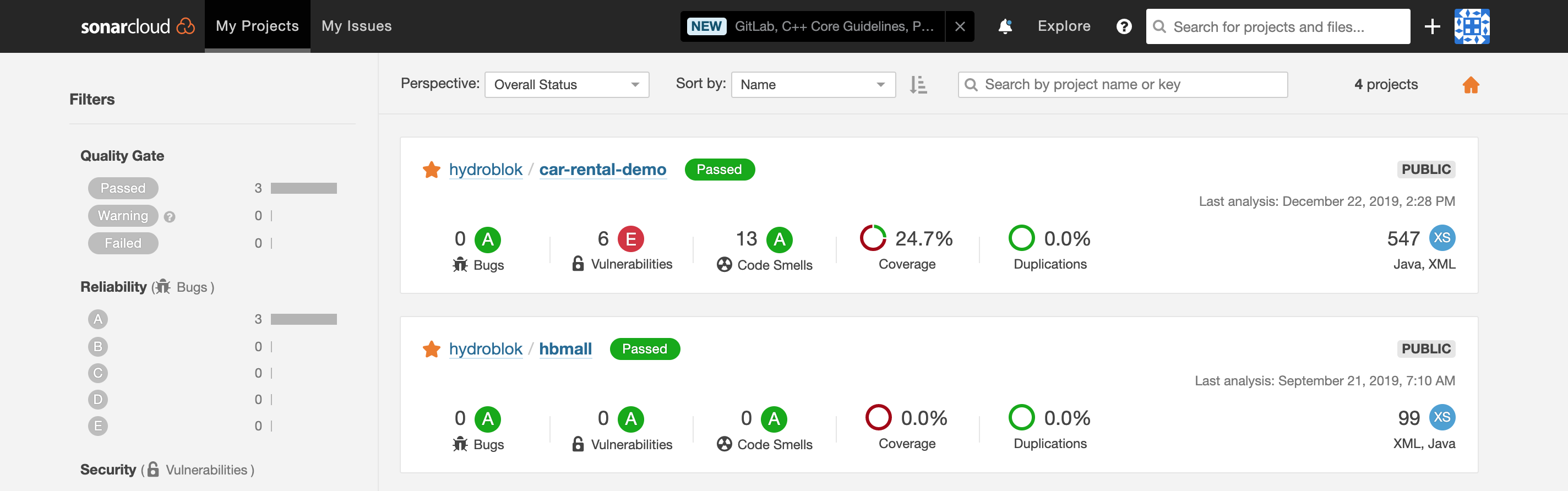




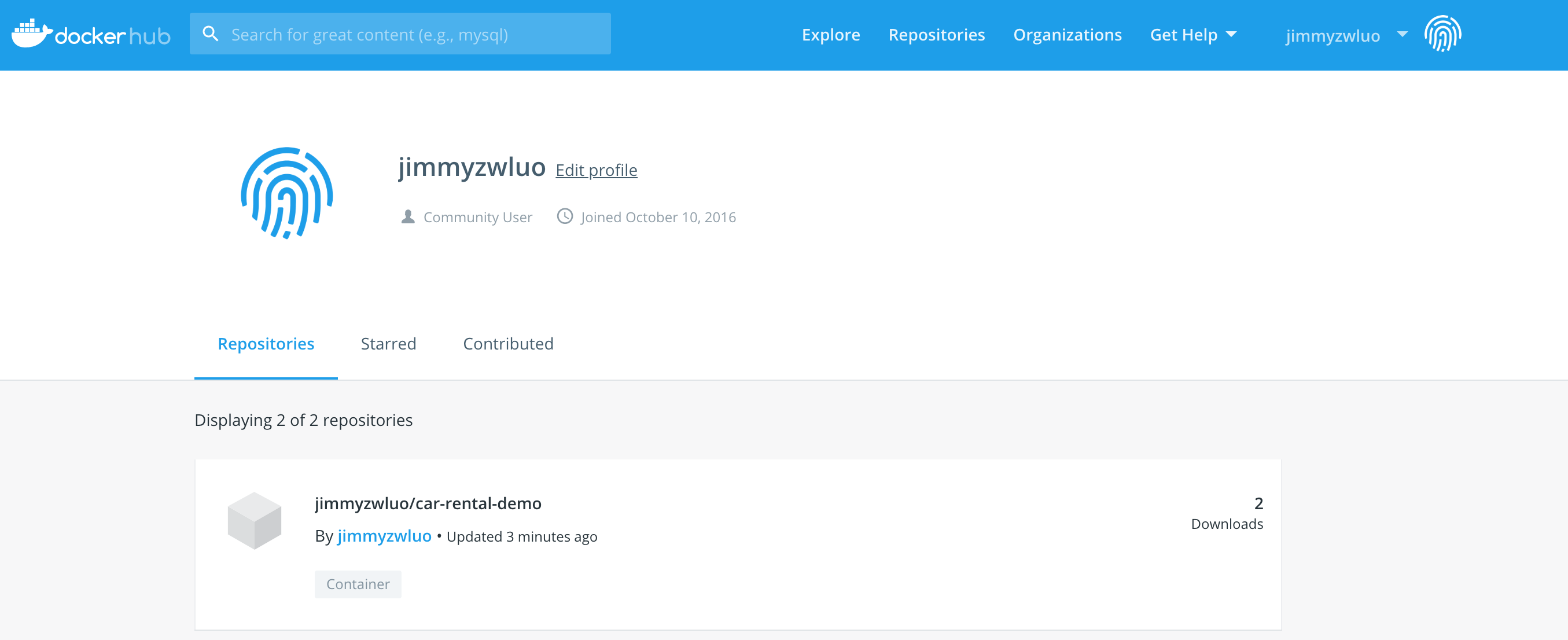
Travis CI



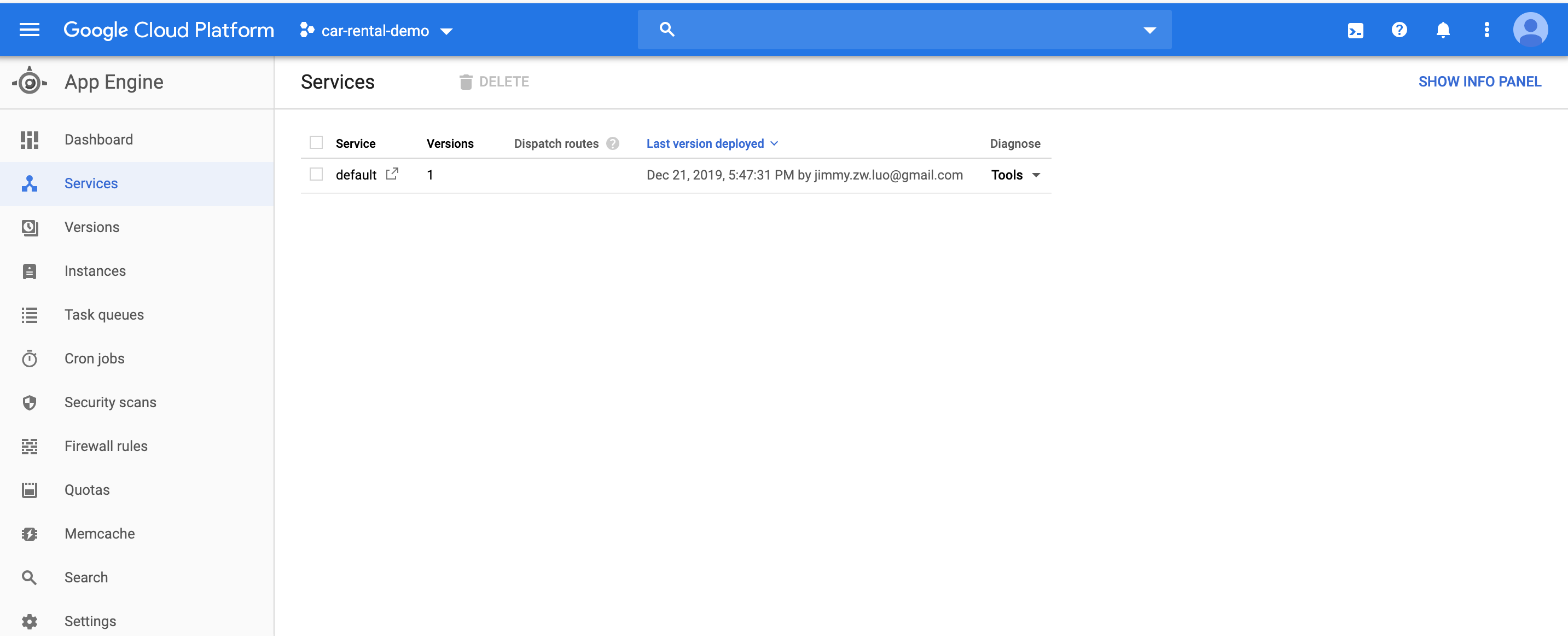
Sonnar Cloud



Docker Hub



Google Cloud Engine

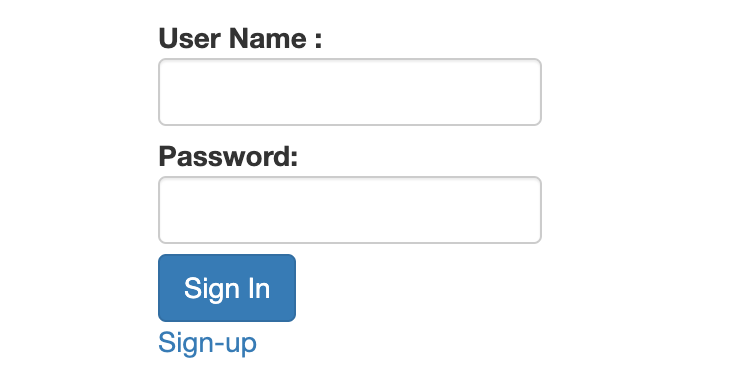


## Google Cloud App Link

As I have use the Azure trial quota before, I can’t deploy the program to Azure, so I deploy it to google cloud. Please use vpn to login the url if you are in China. I have created an account (user name: admin, password: admin) for this demo, you can also sign up a new user for the test.

<http://car-rental-demo-262702.appspot.com/>

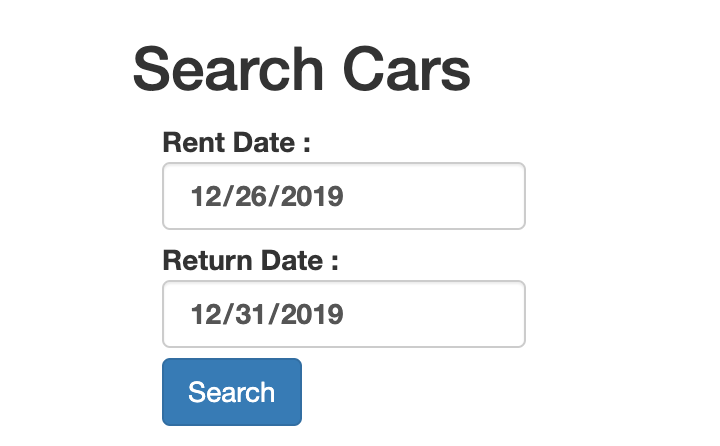
Login page



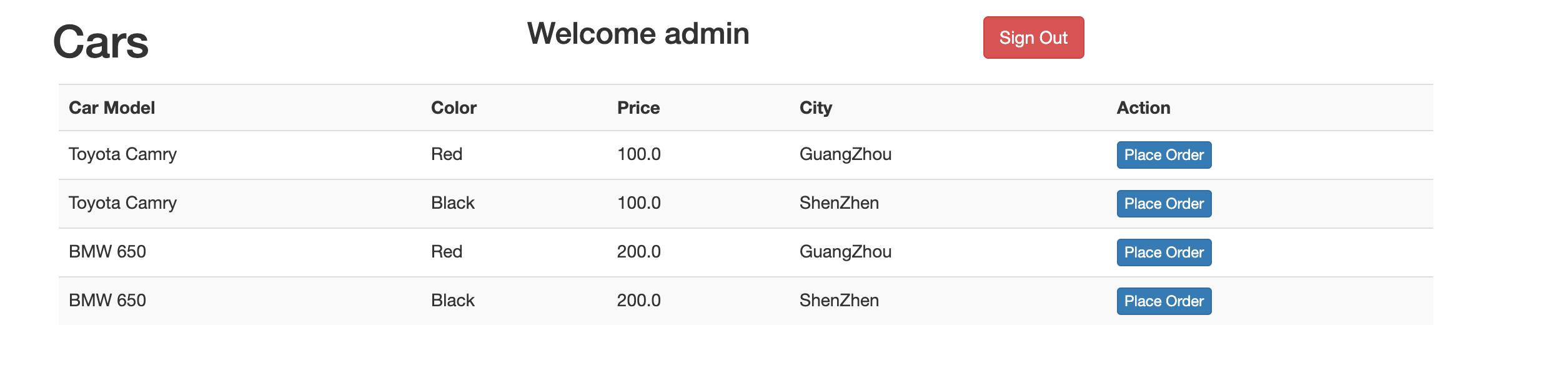
sign up page



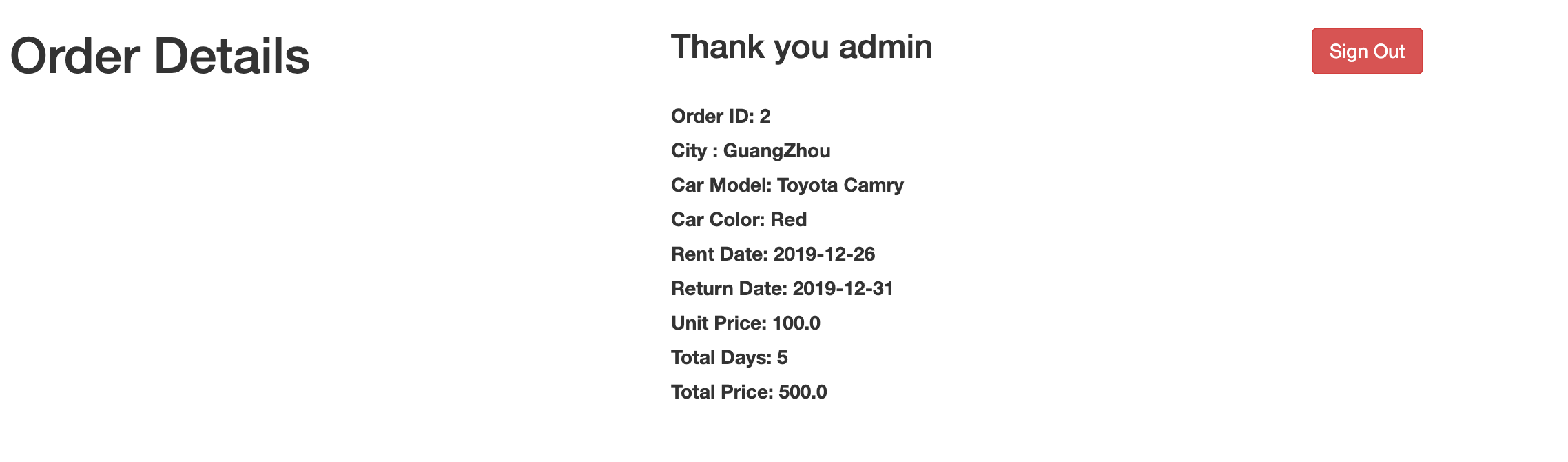
Search page



Select Page:



Order Detail page



## Future Version

When the car rental business runs good, and we have enough resources, we should consider to refactor the system to micro services with using the spring cloud or dubbo framework, and use the middleware like Rabbit MQ, Redis, Zookeeper to setup multiple layers of memory for handling the asynchronous communication and concurrency problems, and we could use the elastic search to solve the global search problem if necessary, we could also use GKE(google cloud engine) or k8s setup in other public cloud to setup the cluster to make our services to be a distributing system, such that our system could be high availability and scalability. Below architecture is just the proposal technology architect for the future version.

