Final Project Part 2

CS – GY 6083 -B

Group Member 1 Name: Fangzhou Jiang ID: fj566

Group Member 2 Name: Yawen Huang ID: yh3288

Submission Date:12/20/2020

Directory

Overview

WOW (World on Wheels) is a car rental company, which has a lot of branches around the United States. To raise working efficiency and meet the need for storing huge amount of data, WOW decides to use database system instead of the file system.

For each office location, WOW wants to save their address and phone number. Each office has various classes vehicles, they should be also recorded separately. Staff can set daily rate for each vehicle and record the pickup location, drop off location, pickup date, drop off date, start odometer and the end odometer of each rental service, then an invoice can be created and sent to the customer automatically.

After the invoice is created, the customer should be able to view the invoice and pay the rent. One invoice can be paid multiple times before it's fully paid. Customer can make multiple payment by different method.

WOW has two types of customer, individual and corporate. These two types customer have different information that need to be recorded. For example, corporation customer may need to record the employee ID to know who rents the car on a corporate account. For individual customer, full name, driver license number, insurance company name and insurance policy number need to be provided.

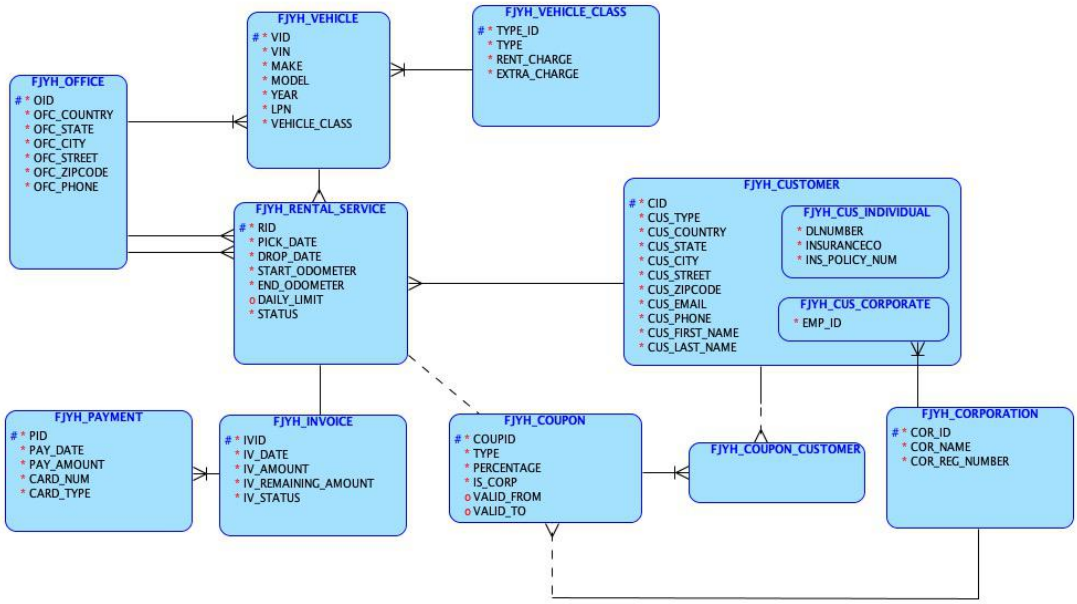
WOW provides coupon for customers occasionally. Customer can have discount by using the coupon. Only one coupon can be used in the rental service. There are two types of coupon: Individual Coupon and Corporation Coupon. For individual coupon, there should be a valid date. Customer can not use the expired coupon. And if it's a corporation coupon, then there's no limit on the coupon but it is linked to a corporation and can only be used by the customer from specific corporation.

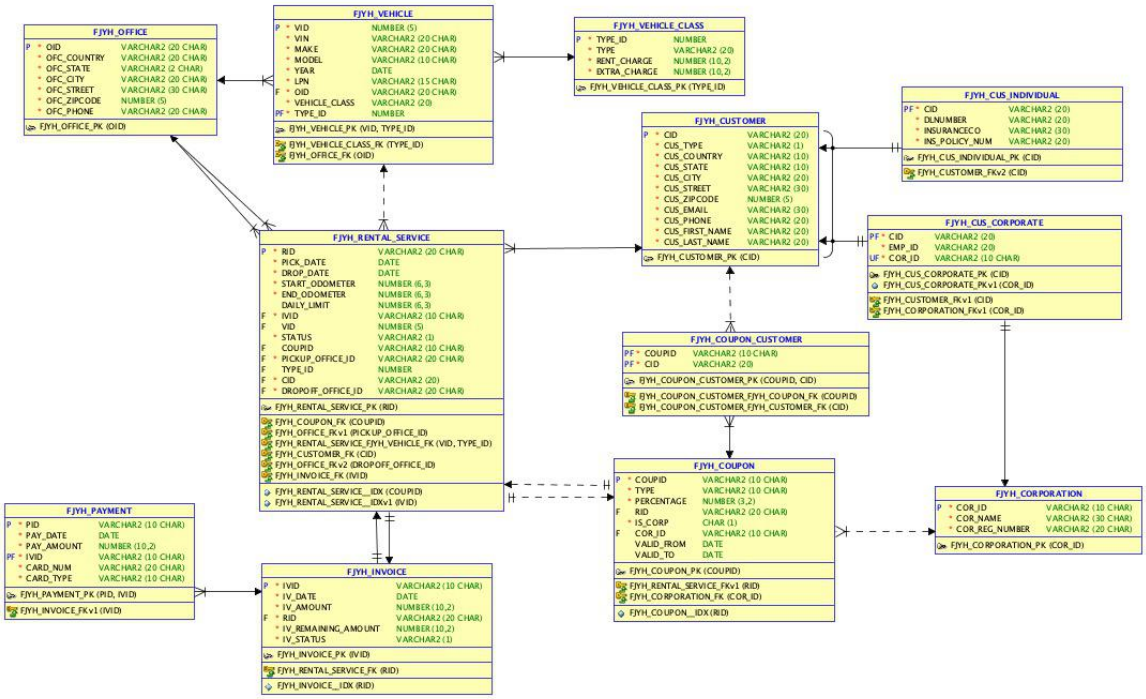
Multiple methods (credit/debit/gift card) can be used in one invoice.

To provide a better resolution towards this business case, we review the whole enterprise modeling components and analyze the overall data requirements of the proposed data information system. We use Data Modeler to design logical and relational model, MySQL for the database and Django framework to connect our backend database. Customer and staff will have different authorization on data, which can protect customers' privacy. We provide a neatly and friendly website for user to perform business activities. By using this business solution, user can retrieve and update data more quickly and conveniently, which will raise the work efficiency a lot and provide customer a better experience.

In this business model, we assume that:

- A corporation can have multiple coupons
- One coupon can be allocated to multiple customers
- One customer can have multiple rental services
- One rental service can only have one invoice





Development Environment

Software: PyCharm

Framework: Django

Language: Python, HTML

Database: MySQL

Number of data

Customer:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_customer`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
13

```
select COUNT(*) from `func_customer` LIMIT 0, 1000
```

Individual Customer:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_individualcustomer`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
9

```
select COUNT(*) from `func_individualcustomer` LIMIT 0, 1000
```

Corporation Customer

Coupon:

Coupon_Customer:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_coupon_customer`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
8

select COUNT(*) from `func_coupon_customer` LIMIT 0, 1000

Corporation:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_corporation`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
5

select COUNT(*) from `func_corporation` LIMIT 0, 1000

Office:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_office`;
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
5

select COUNT(*) from `func_office` LIMIT 0, 1000

Vehicle:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_vehicle`;
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
30

select COUNT(*) from `func_vehicle` LIMIT 0, 1000

Vehicle class:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_vehicleclass`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
6

select COUNT(*) from `func_vehicleclass` LIMIT 0, 1000

Rental service:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_rentalservice`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
12

select COUNT(*) from `func_rentalservice` LIMIT 0, 1000

Invoice:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_invoice`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
9

select COUNT(*) from `func_invoice` LIMIT 0, 1000

Payment:

Query 1 × 历史记录 +

```
1    SELECT COUNT(*) FROM `func_payment`;
```

1 结果 2 个配置文件 3 信息 4 表数据 5 信息

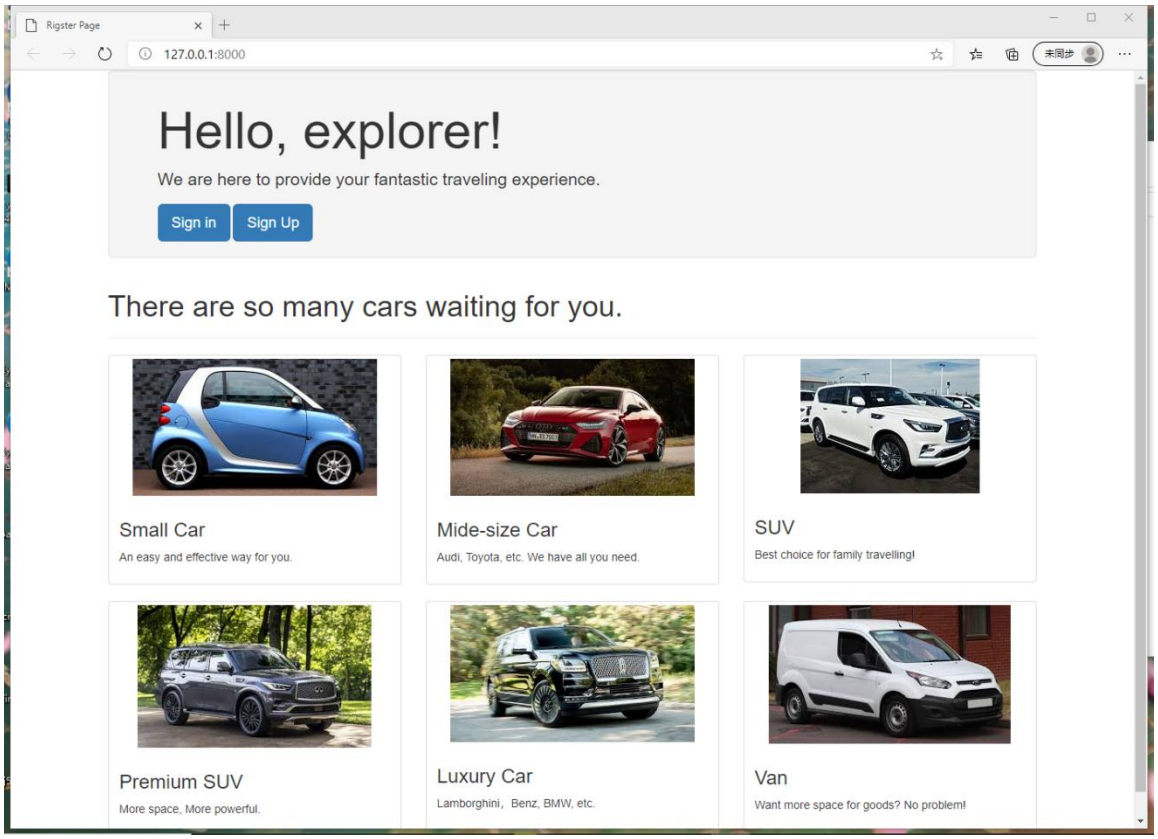
(只读) 限制行 第一行: 0 行数: 1000

COUNT(*)
13

select COUNT(*) from `func_payment` LIMIT 0, 1000

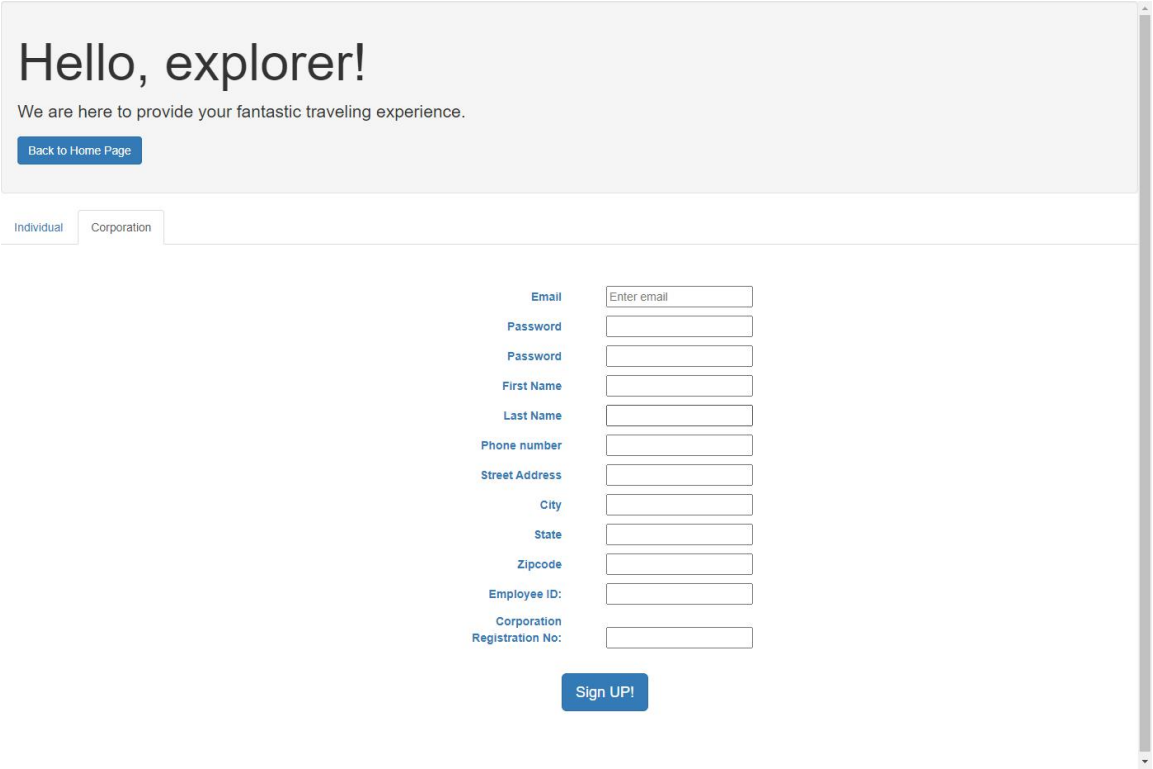
Website overview:

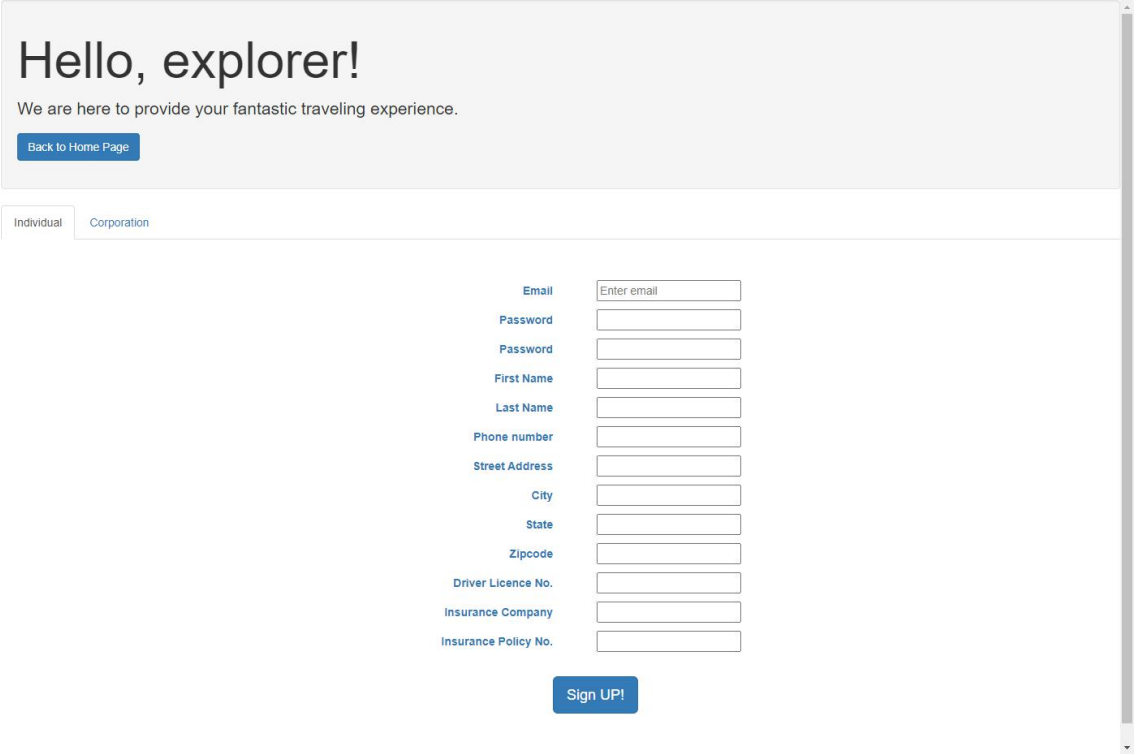
The front page:



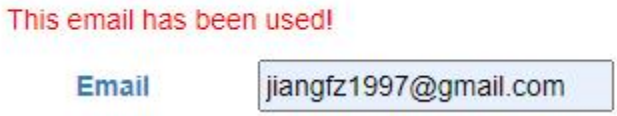
You can sign in or sign up by clicking the buttons.

Sign Up page:

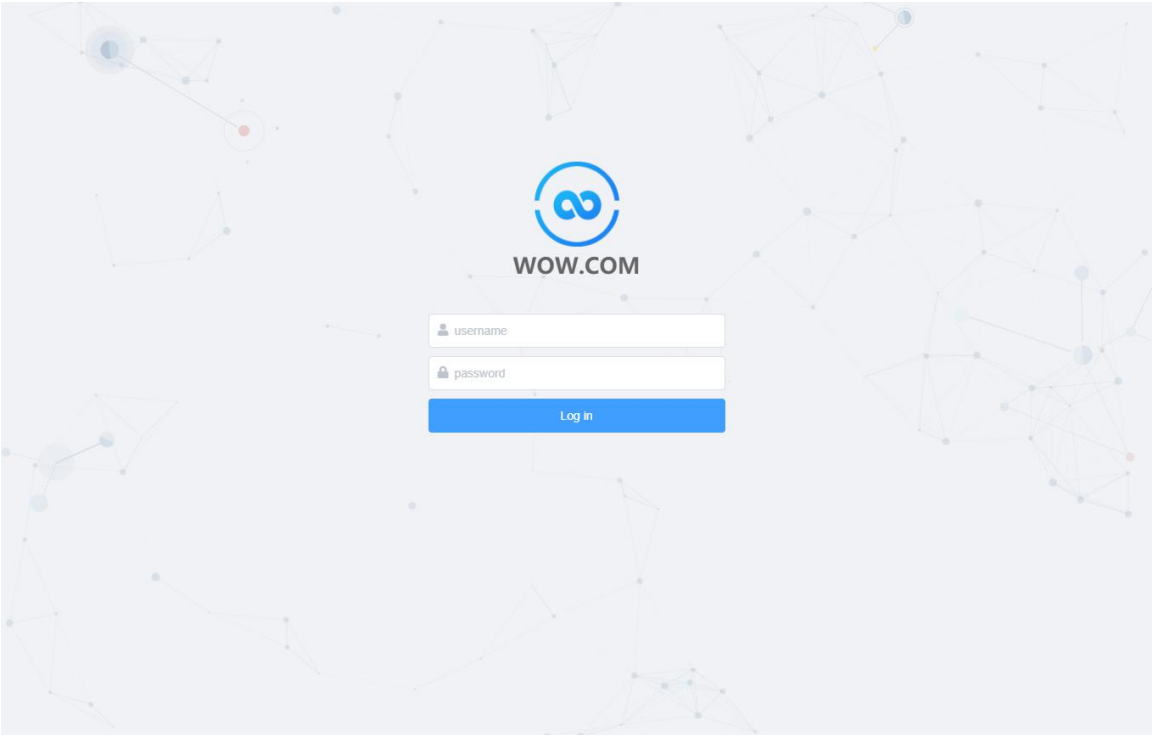

Hello, explorer!
We are here to provide your fantastic traveling experience.
Back to Home Page
Individual
Corporation
Email
Enter email
Password
Password
First Name
Last Name
Phone number
Street Address
City
State
Zipcode
Employee ID:
Corporation Registration No:
Sign UP!


Hello, explorer!
We are here to provide your fantastic traveling experience.
Back to Home Page
Individual
Corporation
Email
Enter email
Password
Password
First Name
Last Name
Phone number
Street Address
City
State
Zipcode
Driver Licence No.
Insurance Company
Insurance Policy No.
Sign UP!

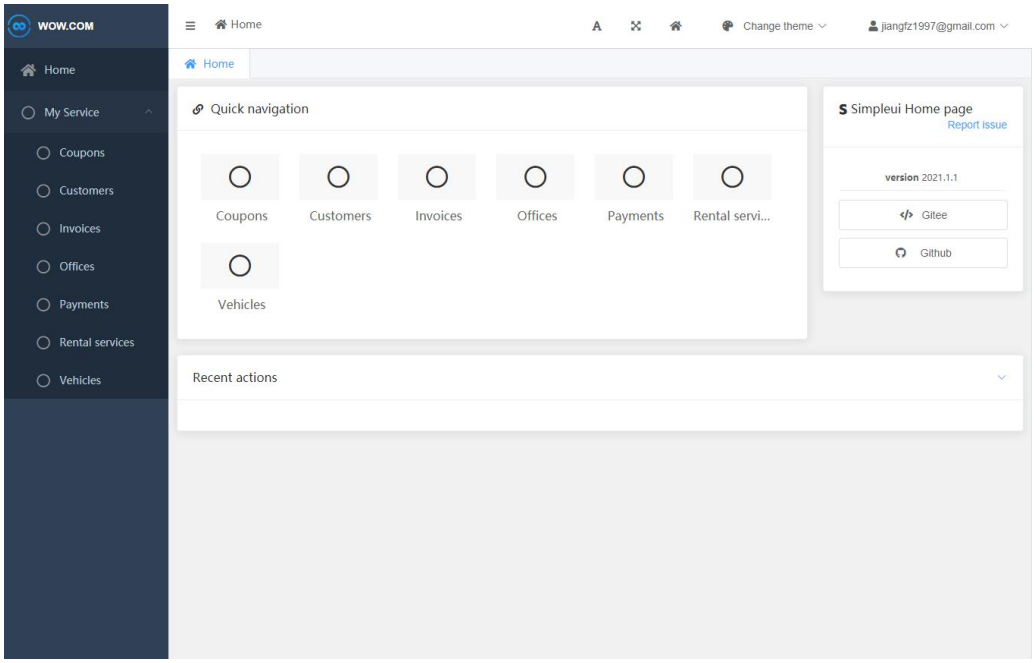
The sign up page is separated into two part: Individual and Corporation. They have some difference between each other. The website can detect the invalid input like invalid email, email which is already signed up, two different password and blank information. For corporation customer, it can also detect if the corporation number is valid. Notice that the employee's account should be created by the root user, not by themself.



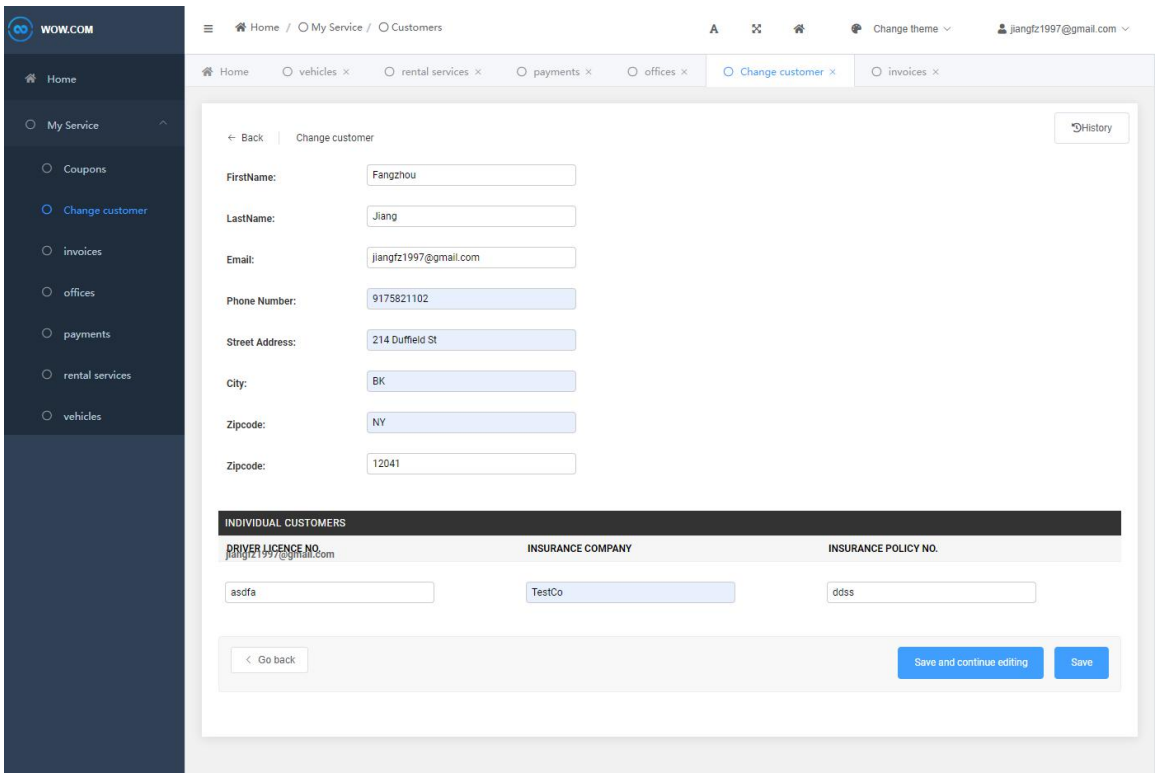
After sign up, we can sign in as a customer or employee in sign in page:



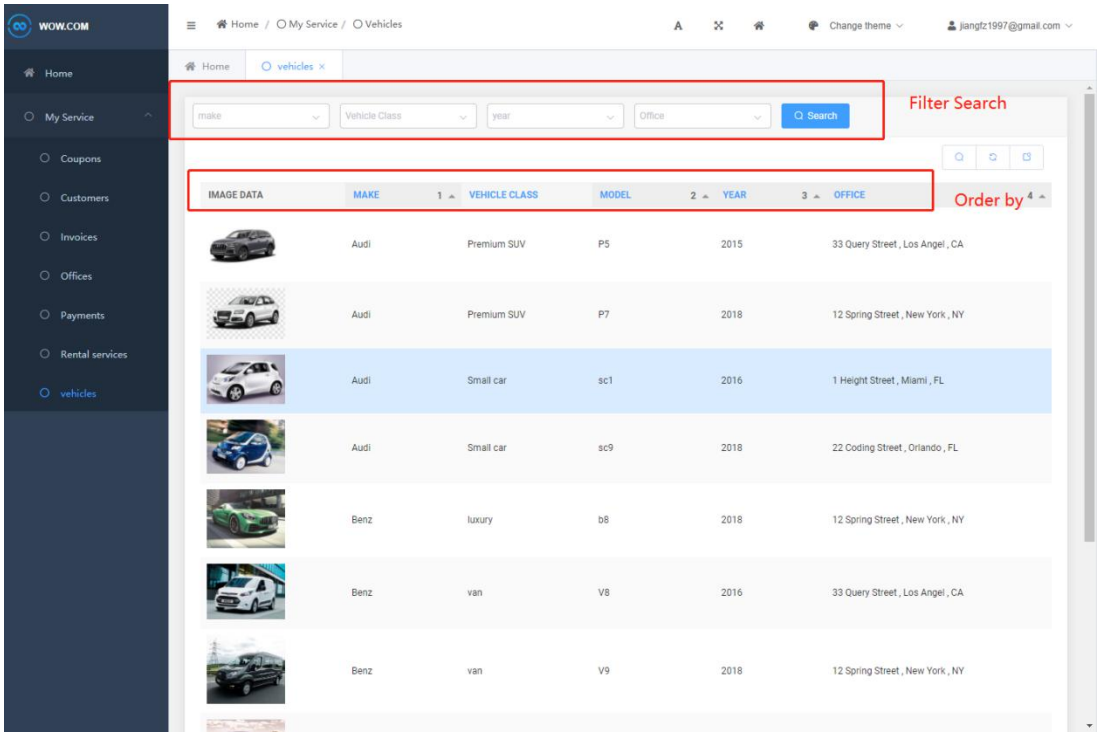
After sign in, we got into the user interface:



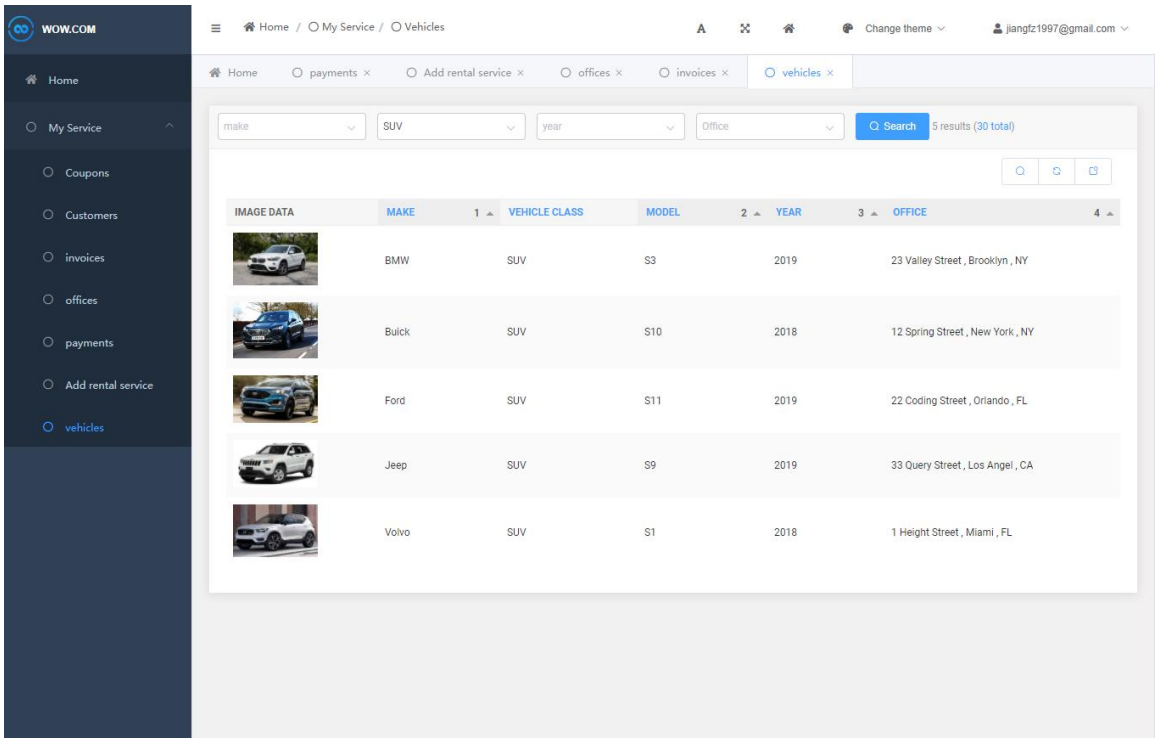
As a customer, we can check and modify our personal information:



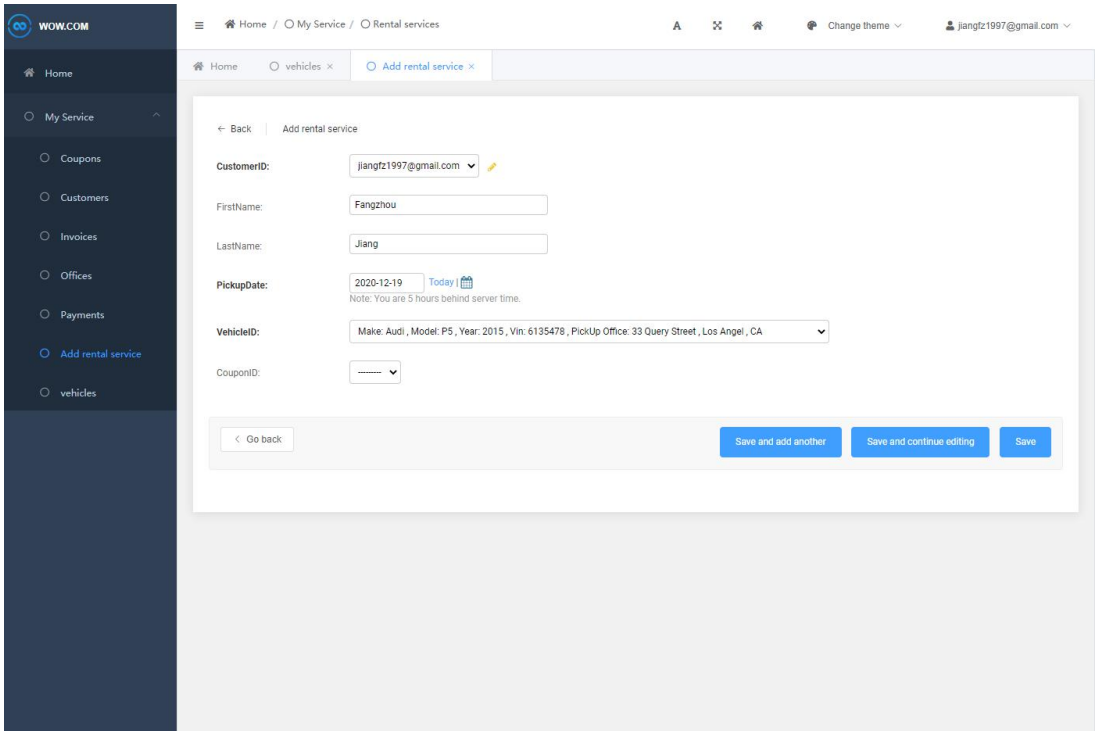
Browse the vehicle:



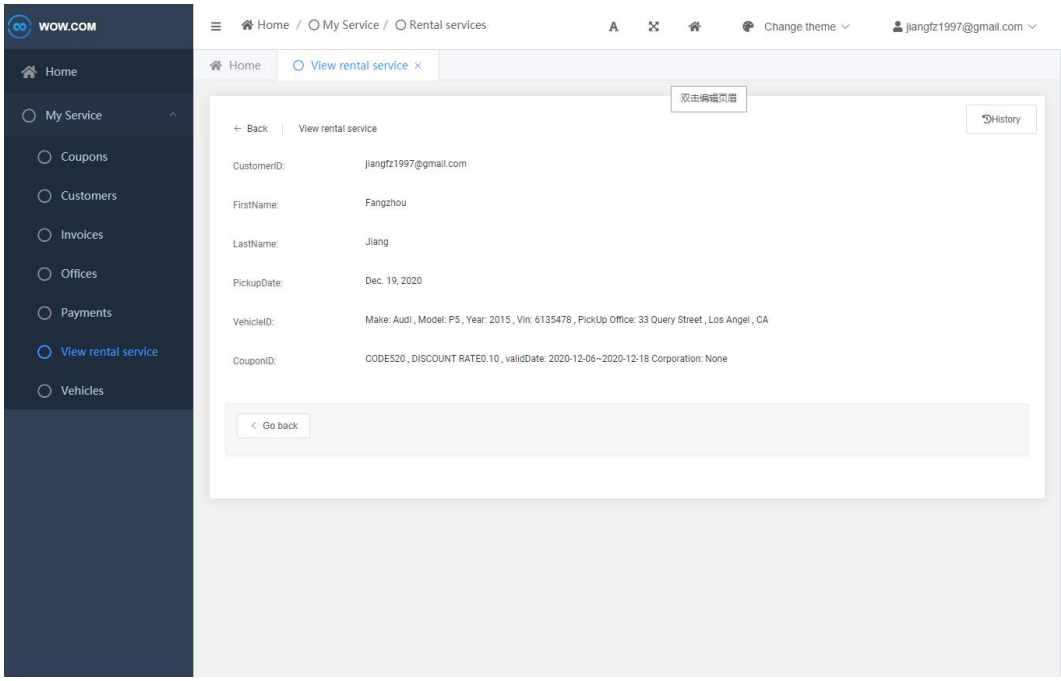
We can use the search box to search car by make, class, year and location.



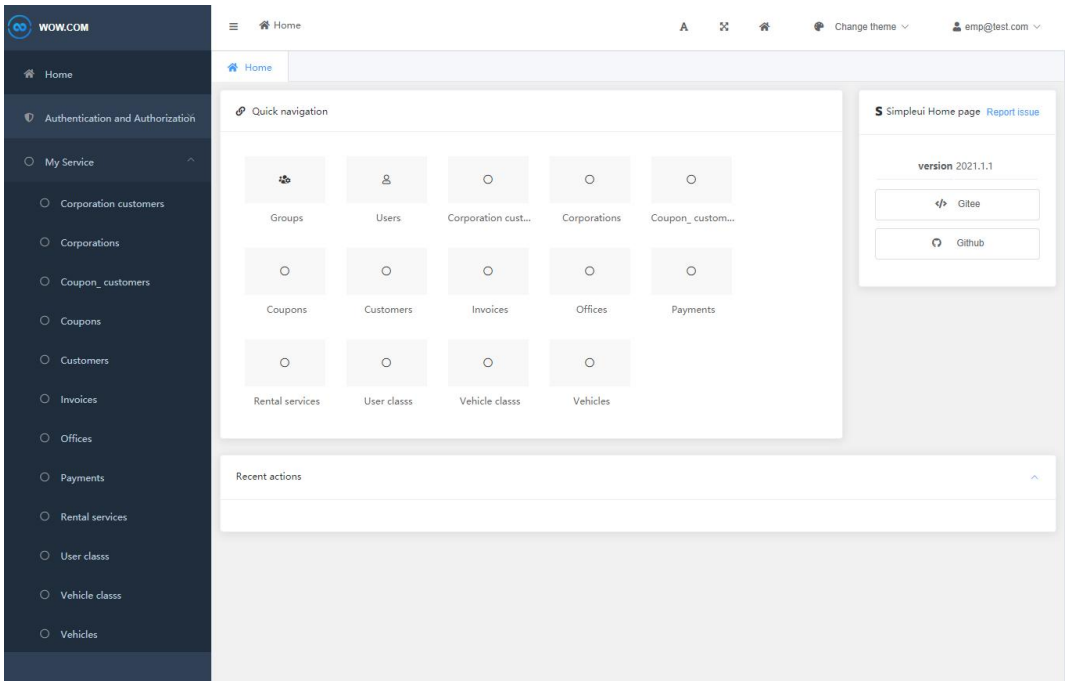
After finding a perfect car, we can go to the rental service page to submit an order, notice that as a customer the thing we can modify is very limited:



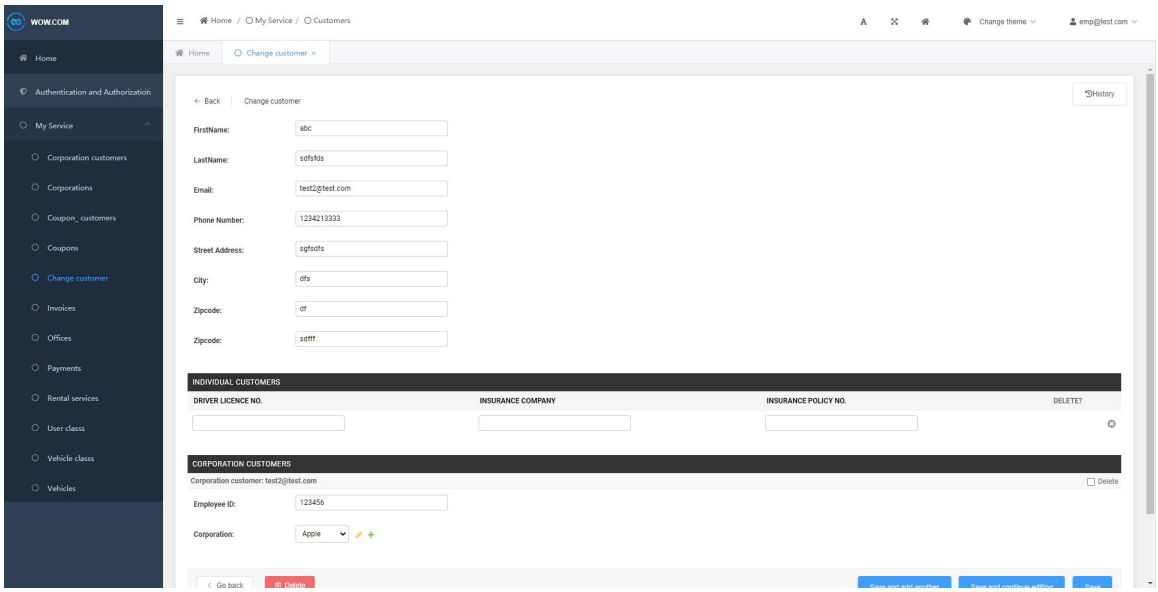
After making the rental service, we cannot modify it anymore, we can only view it.

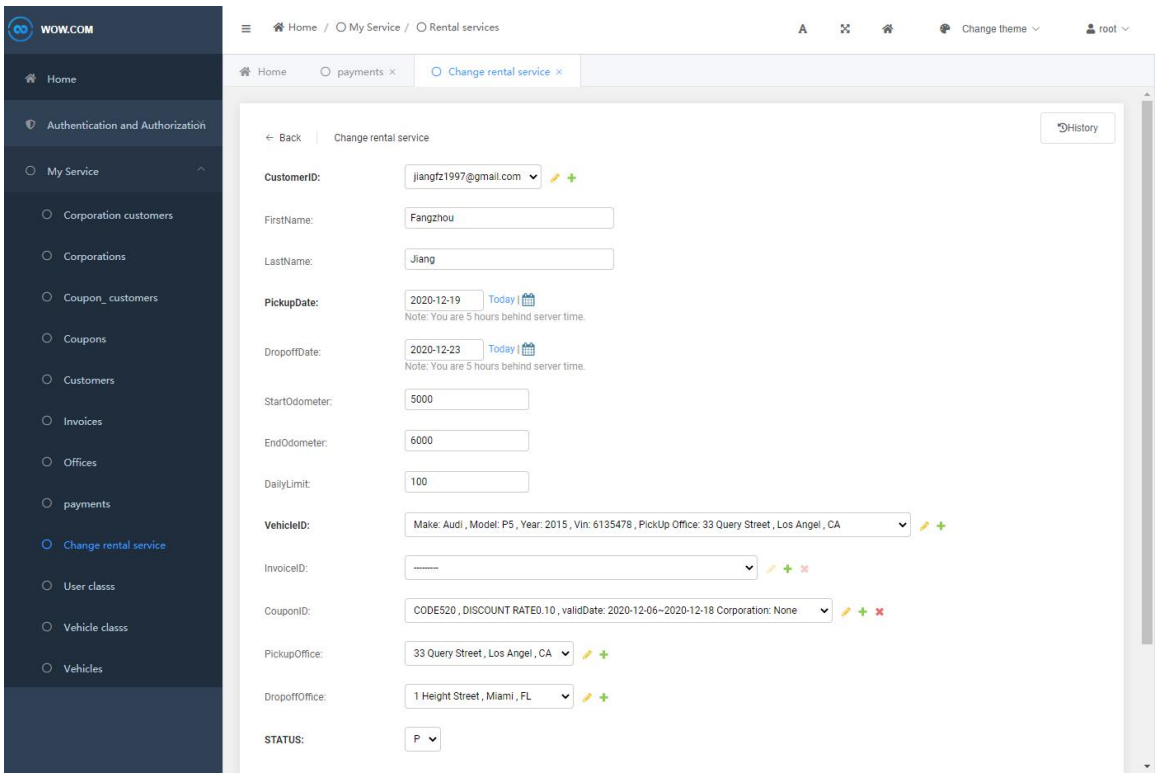


Then we go to the employee's account to fulfill the information. In our website, the rental service is completed by the employee because the customer would not know about some information like End Odometer. So the employee will fill up the rest of the form after the car is returned.

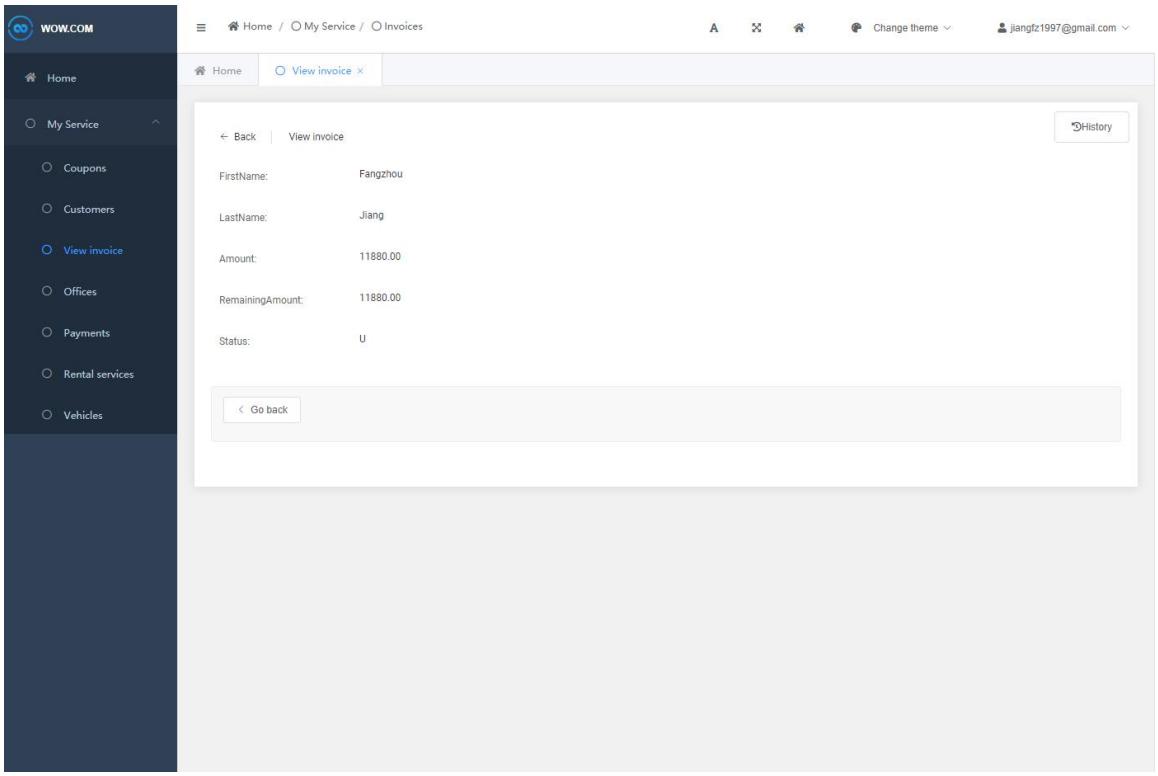


As an employee, we can modify all the informations except the user's password.



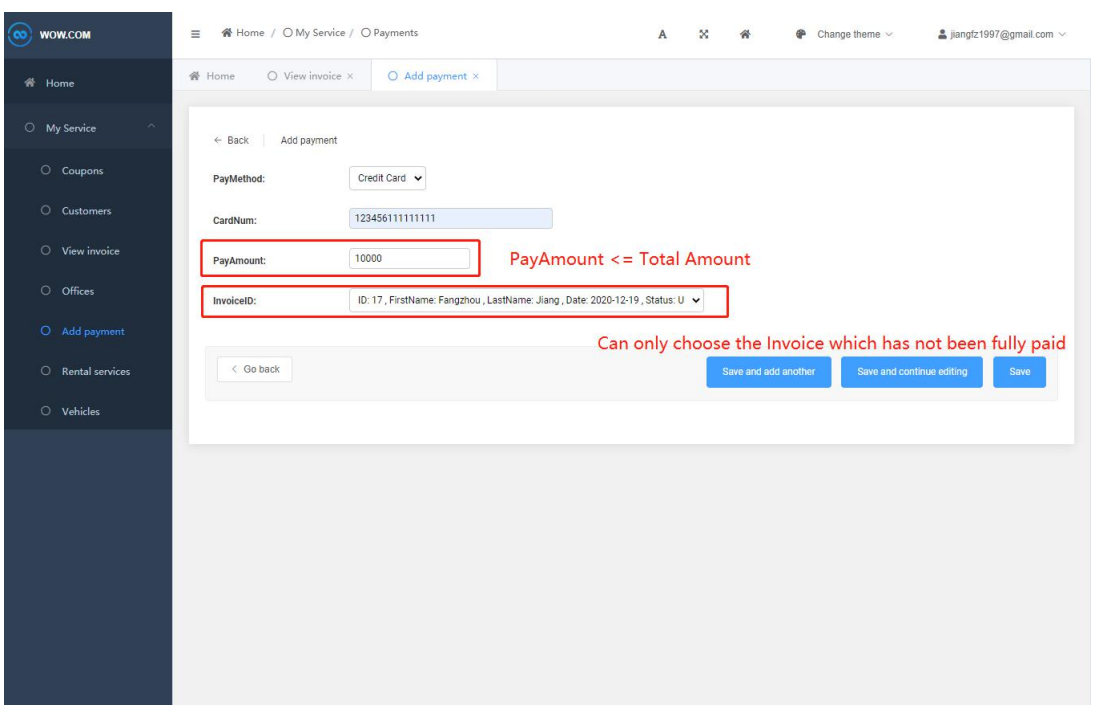


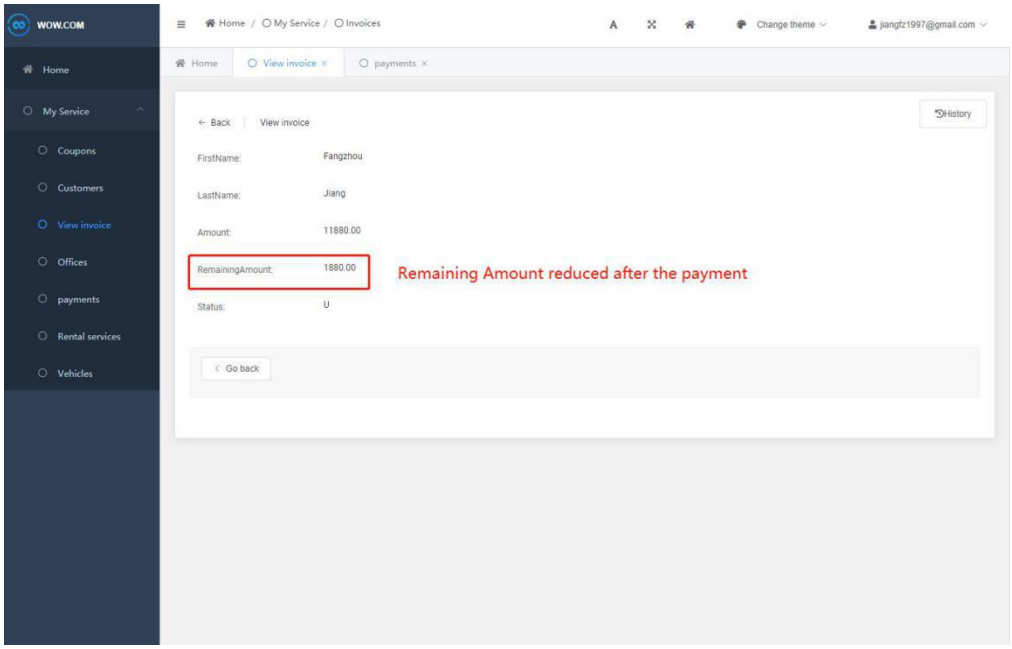
After the rental service has complete, an invoice will be created automatically. The Amount is the total fee calculated automatically.



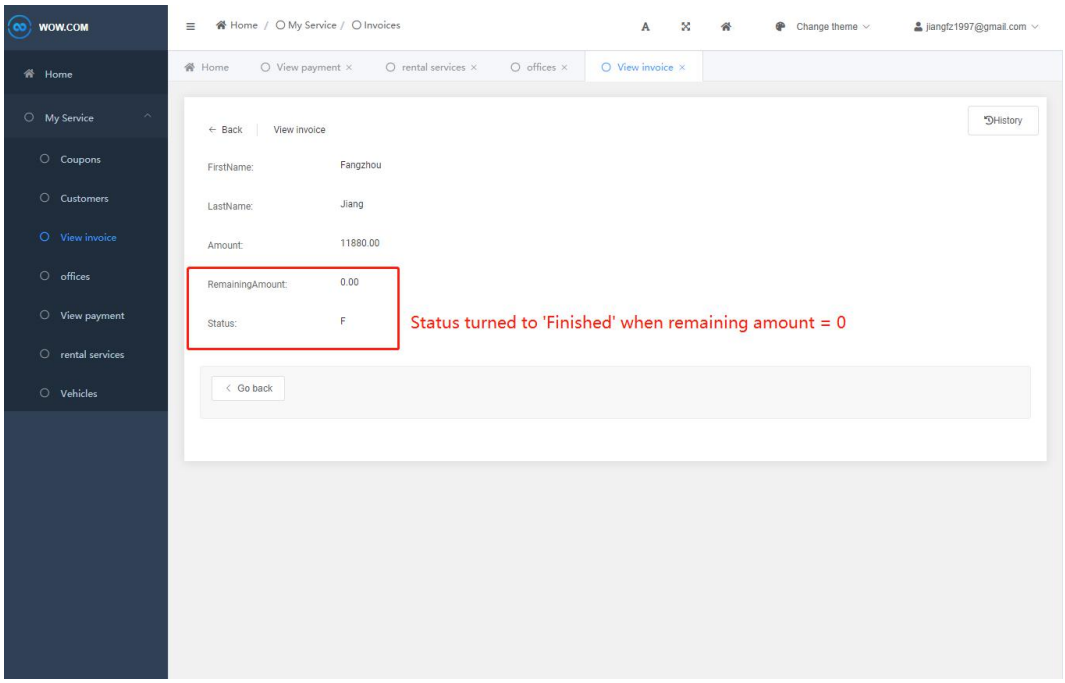
After that, the user can see the invoice and make payment to it. User can make multiple payment for one invoice by different method.

In this case, the customer paid 10000$ first, we can see the remaining amount changed.

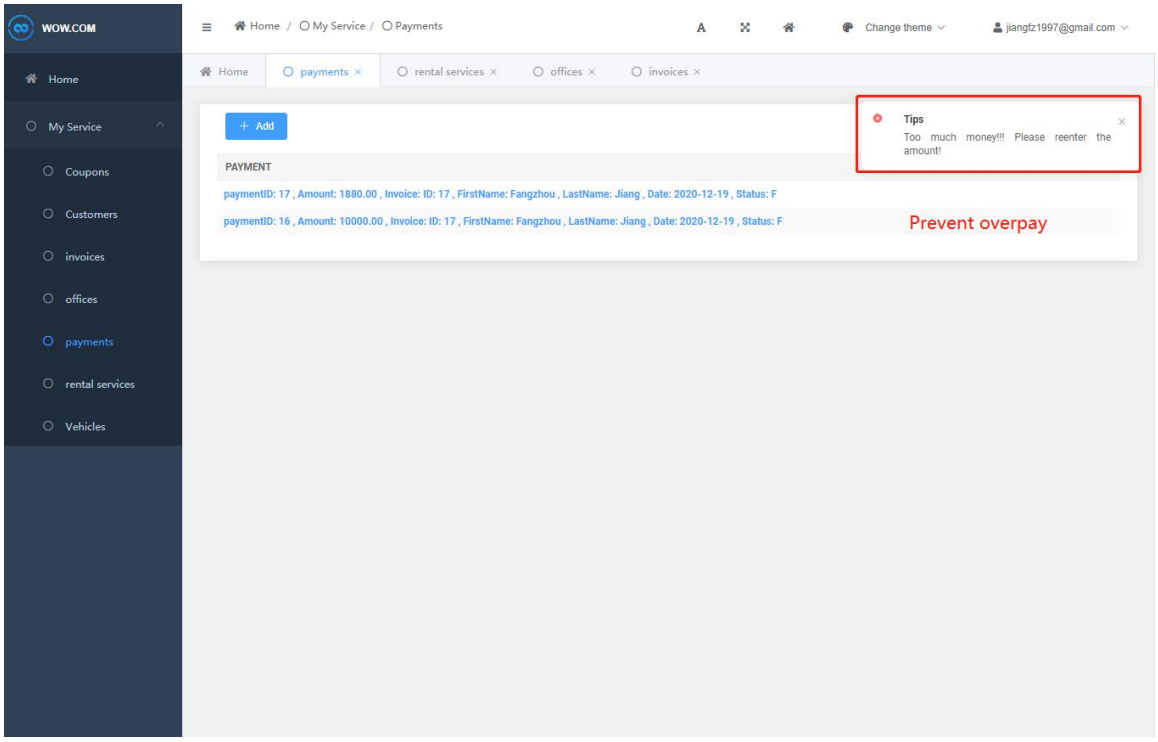




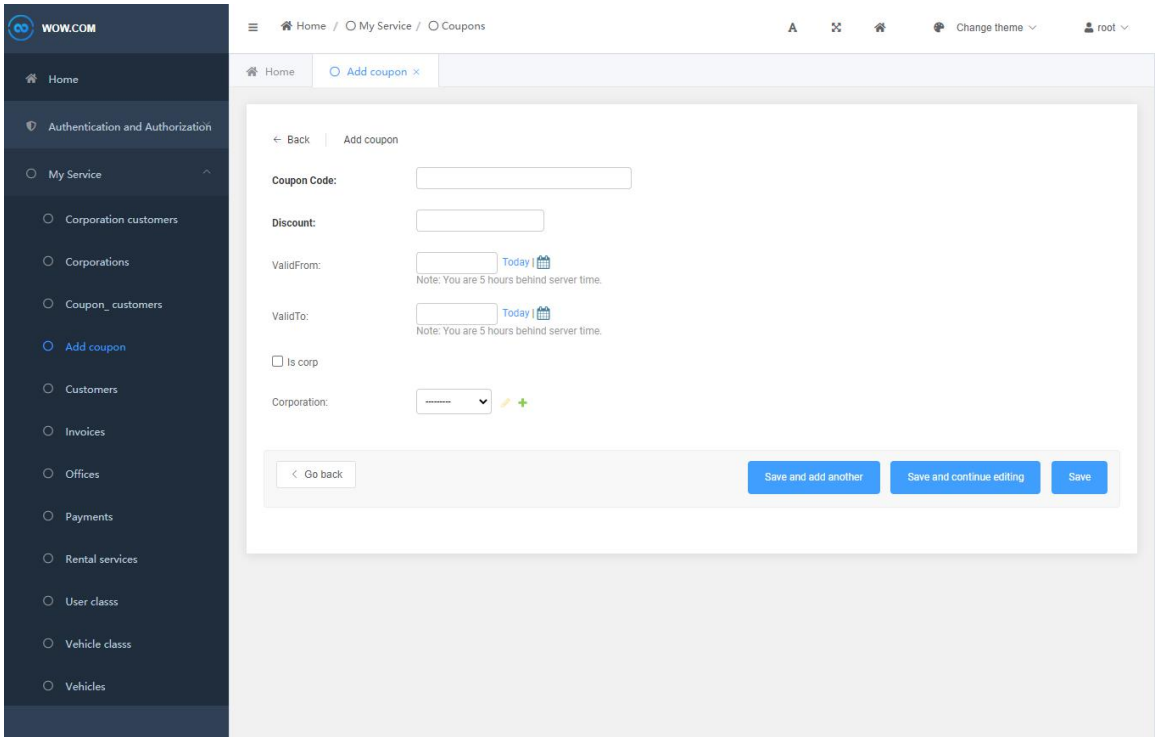
The customer paid the rest of amount, when the remaining amount is 0, the invoice is finished.

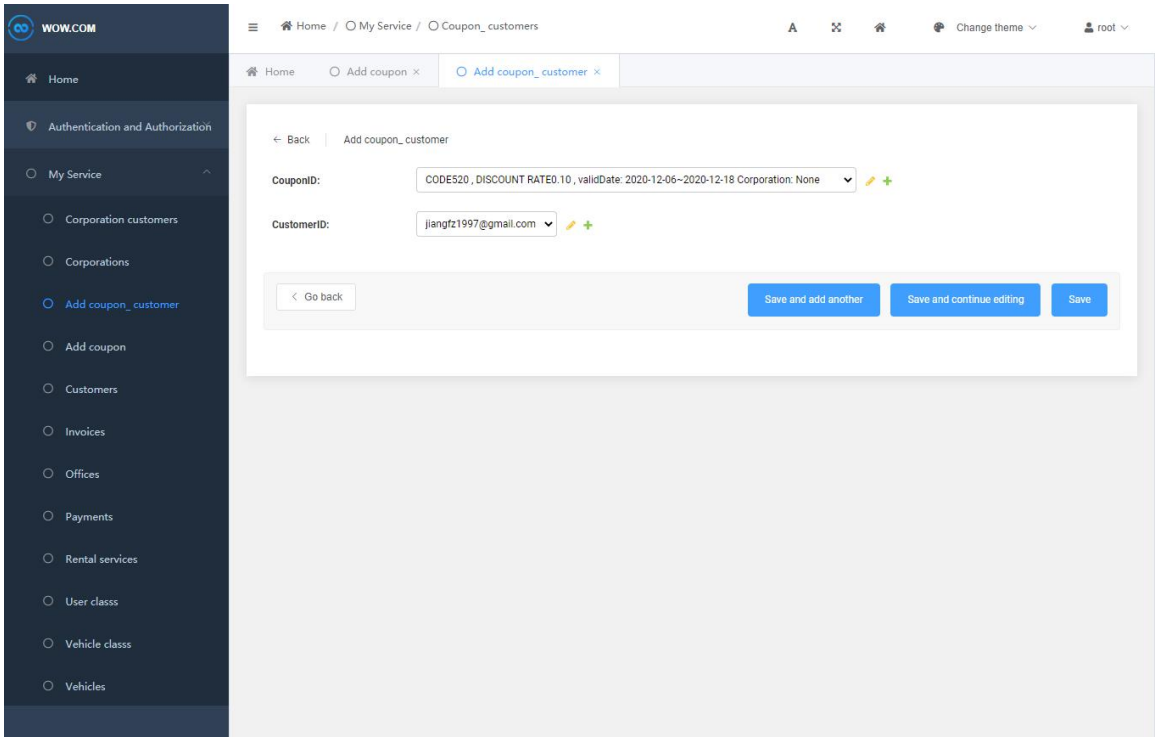


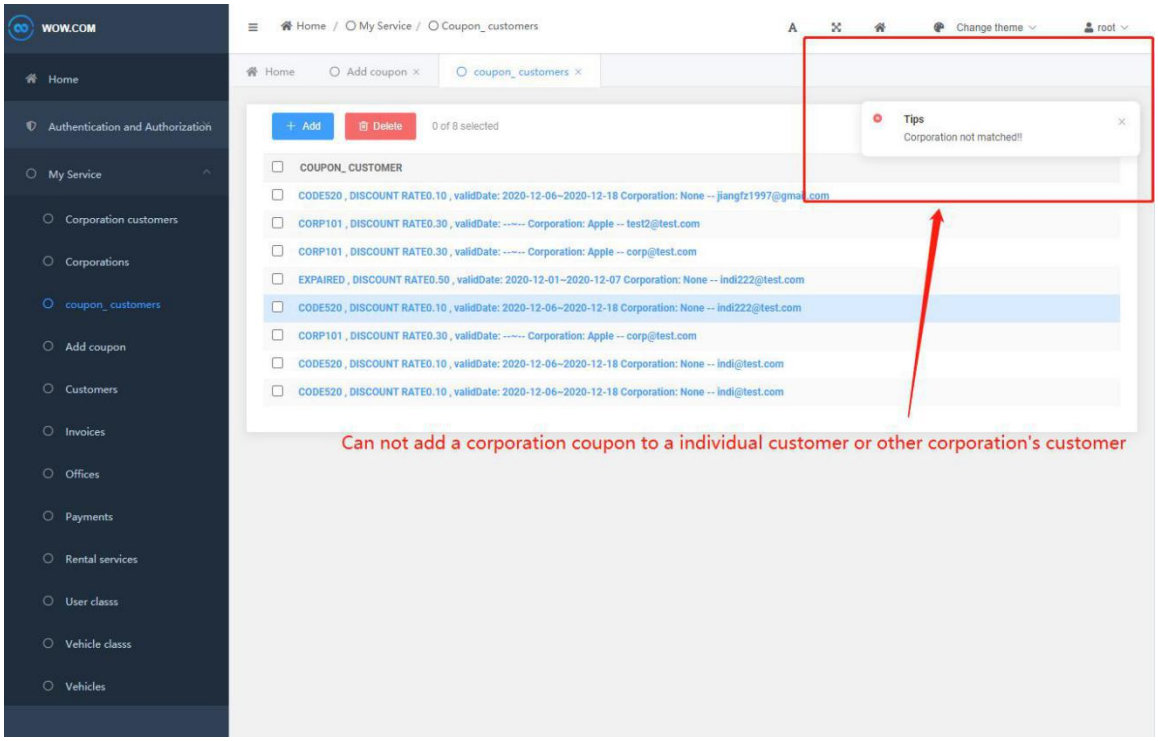
Notice that the customer can not overpay the invoice, and they can not pay for a finished invoice.



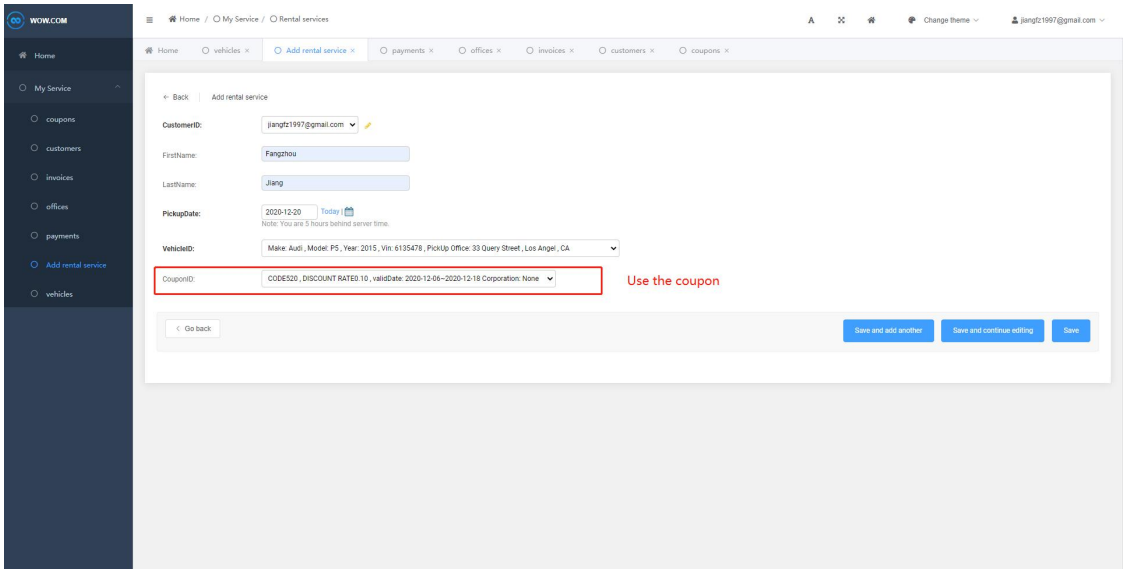
About the coupon, the employee can add a coupon. If the coupon is a corporation coupon, then it does not have the time limit and can only be distributed to the employees of the specific company.

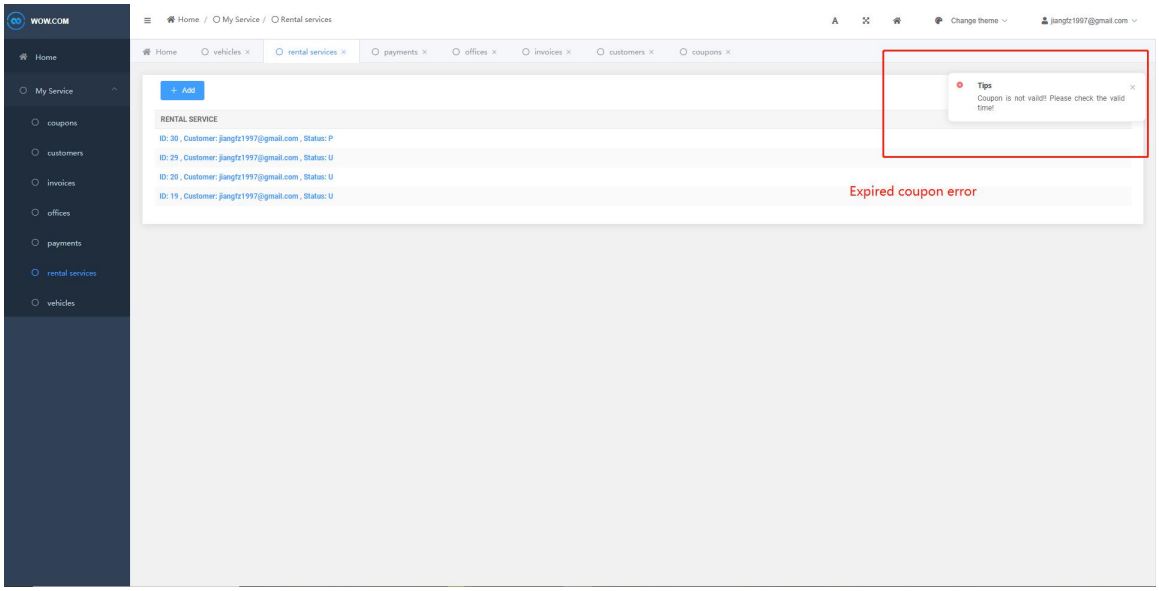



WOW.COM
Home / My Service / Coupon_ customers
Change theme
root
Home
Authentication and Authorization
My Service
Corporation customers
Corporations
Add coupon_ customer
Add coupon
Customers
Invoices
Offices
Payments
Rental services
User classs
Vehicle classs
Vehicles
Home
Add coupon
Add coupon_ customer
Back
Add coupon_ customer
CouponID:
CODE520 , DISCOUNT RATE0.10 , validDate: 2020-12-06~2020-12-18 Corporation: None
CustomerID:
jiangfz1997@gmail.com
Go back
Save and add another
Save and continue editing
Save


WOW.COM
Home / My Service / Coupon_ customers
Change theme
root
Home
Authentication and Authorization
My Service
Corporation customers
Corporations
coupon_ customers
Add coupon
Customers
Invoices
Offices
Payments
Rental services
User classs
Vehicle classs
Vehicles
Home
Add coupon
coupon_ customers
Add
Delete
0 of 8 selected
Tips
Corporation not matched!!
COUPON_ CUSTOMER
CODE520 , DISCOUNT RATE0.10 , validDate: 2020-12-06~2020-12-18 Corporation: None -- jiangfz1997@gmail.com
CORP101 , DISCOUNT RATE0.30 , validDate: --~-- Corporation: Apple -- test2@test.com
CORP101 , DISCOUNT RATE0.30 , validDate: --~-- Corporation: Apple -- corp@test.com
EXPAIRED , DISCOUNT RATE0.50 , validDate: 2020-12-01~2020-12-07 Corporation: None -- indi222@test.com
CODE520 , DISCOUNT RATE0.10 , validDate: 2020-12-06~2020-12-18 Corporation: None -- indi222@test.com
CORP101 , DISCOUNT RATE0.30 , validDate: --~-- Corporation: Apple -- corp@test.com
CODE520 , DISCOUNT RATE0.10 , validDate: 2020-12-06~2020-12-18 Corporation: None -- indi@test.com
CODE520 , DISCOUNT RATE0.10 , validDate: 2020-12-06~2020-12-18 Corporation: None -- indi@test.com
Can not add a corporation coupon to a individual customer or other corporation's customer

Customer can not use the expired coupon.





For extra feather, we used index for the vehicle search. Since we have four filters: MAKE, CLASS, YEAR, LOCATION, we designed the index in django's model like this:

```
class Meta:
    indexes = [models.Index(fields=['make', 'model', 'year', 'locationID'])]
```

We also worked on the security part, which will be explained as follow:

For the security protection, we did it through several ways.

Password security:

First of all, we store the user's password after encryption. Instead of storing the User ID and password in the Customer table, we stored it in the user table provided by the Django Framework. And use the create_user, login() and logout() API function from Django API to achieve the user's account operation.

```
user = User.objects.create_user(username=email, password=password1, is_staff=True)
user.groups.add(2)
logout(request)
```

login(request, user)

And after the user has been created, the password will be encrypted by Django using hash function.

id	password	last_login
2	pbkdf2_sha256$216000$R5jiEsGguUqL$eWyEH5FugDG4/MgRKDHOB6S3cWR3Y2gul	2020-12-17 19:25:29.821559
24	pbkdf2_sha256$216000$lwSMS3xR6cpu$rUwnyPInJS2kr0q54EdC8pntdcuyzUcB3	2020-12-17 19:35:47.660086
25	pbkdf2_sha256$216000$ERGlMsxfBQVZ$tKglyMmi8thpeeKz074USCF9rn+dM+zQY	2020-12-17 19:38:18.977401
26	pbkdf2_sha256$216000$kcpZ3wlEx2Hh$MbqVADKeH2tk00jUGnYcY3Vr7sJSoR15z	2020-12-17 20:28:23.202125
27	pbkdf2_sha256$216000$0JdPkMncQKHO$f6LWSKH/2/0NpXbOVO7cPYEJbSGG2Wb2/	2020-12-17 06:46:17.003071
28	pbkdf2_sha256$216000$aUo0uqPxvR8S$3qV6RRSlm/+KqPy3MYYqIDmlwx7MFZ8gm	2020-12-17 06:47:23.360137
29	pbkdf2_sha256$216000$lgPWMvKl27YX$29bSSvuqomnLkUXRwS0c8+4bPNry3YKZ2	2020-12-17 06:47:07.332638
30	pbkdf2_sha256$216000$Xfbg2HsgqmoA$4oQ2nOh6X0U9b9wnyQlo4MVYBcwlcPvdl	2020-12-17 07:01:44.651779
31	pbkdf2_sha256$216000$xoN5AnK8cbuB$EP5CHMpvvg0IhJD5gHxfq0/0lwif+xhXH	2020-12-17 18:38:57.724697
32	pbkdf2_sha256$216000$to3uioHWbgDI$tolCRo2qyCazU72hlaUNIax+Imu3ON+Er	2020-12-17 19:13:51.463988
33	pbkdf2_sha256$216000$ua5ZYlb6YoGs$2+X/3HZNEiqB5bgsQkXlPsOaqruO6lWNG	2020-12-17 18:59:41.460964
34	pbkdf2_sha256$216000$uo7eZQgNC3OD$r5aG5aloyFKtcIeYM6MNtkHY2F00Fihzw	2020-12-17 19:09:12.723760
35	pbkdf2_sha256$216000$IeHNpGKHjiLz$5YlGLXKWArc6YQFmpJ/ikt0/9Xxy0Fc96	2020-12-17 19:29:31.101515
36	pbkdf2_sha256$216000$VS5tZCmrFb8L$lMq8ONZev2FqaOUz86FiOwGFOkkvlvfDC	2020-12-17 20:23:26.096545

Transaction:

Then, we need to deal with the transaction. Since there might be multiple queries in one transaction, we need to meet the ACID principle. For doing that, here's an example:

```
cust = Customer(emailID=email, firstName=firstName, lastName=lastName, phoneNo=phoneNo,
streetAddr=streetAddr,city=city, state=state, zipcode=zipcode, customerType='I')
indiU = individualCustomer(customerID=cust, driverLicenceNo=DLNO,
insuranceCompany=insuranceCompany, insurancePolicyNo=insPolicyNo)
try:    #Transaction begin
    with transaction.atomic():
        cust.save()
        indiU.save()
```

```
        messages.error(request, 'Success!')
          user = User.objects.create_user(username=email, password=password1, is_staff=True)
          user.groups.add(3)
          logout(request)
          login(request, user)
          return redirect("/admin")
except DatabaseError:    #Rollback
    messages.error(request, 'Failed! Please check your information!')
    return redirect("/register")
```

In this transaction, a user has been created. As I mentioned before, The password is separated from the Customer table. And the individualCustomer will also be created. So there are three objects being created and save into the database. By doing the transaction.atomic(), all the objects will be saved into the database if there's no problem, but when there is an error, all three objects will roll back.

And to prevent the customer to register with some false information, I added some check in the registration process. Like checking if the email is valid, password and re-entered password are matched, no blank input etc.

Permission

In order to prevent the user from obtaining permissions that do not belong to him, we created three group: Individual Customer, Corporation Customer and Employee. Each class have different permissions on different resources. For example, in the rental service part, Customer should only be able to add or view a service, but not change them after the rental service has been made.

And the employee should have all the permissions except view or change customer's password.

Sql Injection

Then for preventing the sql injection, we used the Objective Relational Mapper, which is a in-build API provided by Django. Django then converts the Python query to SQL query and communicates with the database.

A simple select query is like this:

```
cust = Customer.objects.filter(emailID=request.user.username).first()
```

CSRF attack

For csrf attack, we use csrf token in the html. We also used form to send POST request to the back end.

```
<form action="/registerIndi/" class="form-inline" style="margin-left : 430px"
method="post">{% csrf_token %}
```

Summary:

Since this is the first time that we create a website totally by ourselves, this project really taught us a lot. The project helps us have a better understanding on the Database and more familiar to the operations like creating triggers, procedures etc. We also learned a new programming language Python. By using the framework Django to connect the database and the front end, we got a full view on the website development, learned more about how the website we browse every day works. And during the debugging, we learned not only the solution of the problem, but also the way to solve the problem.

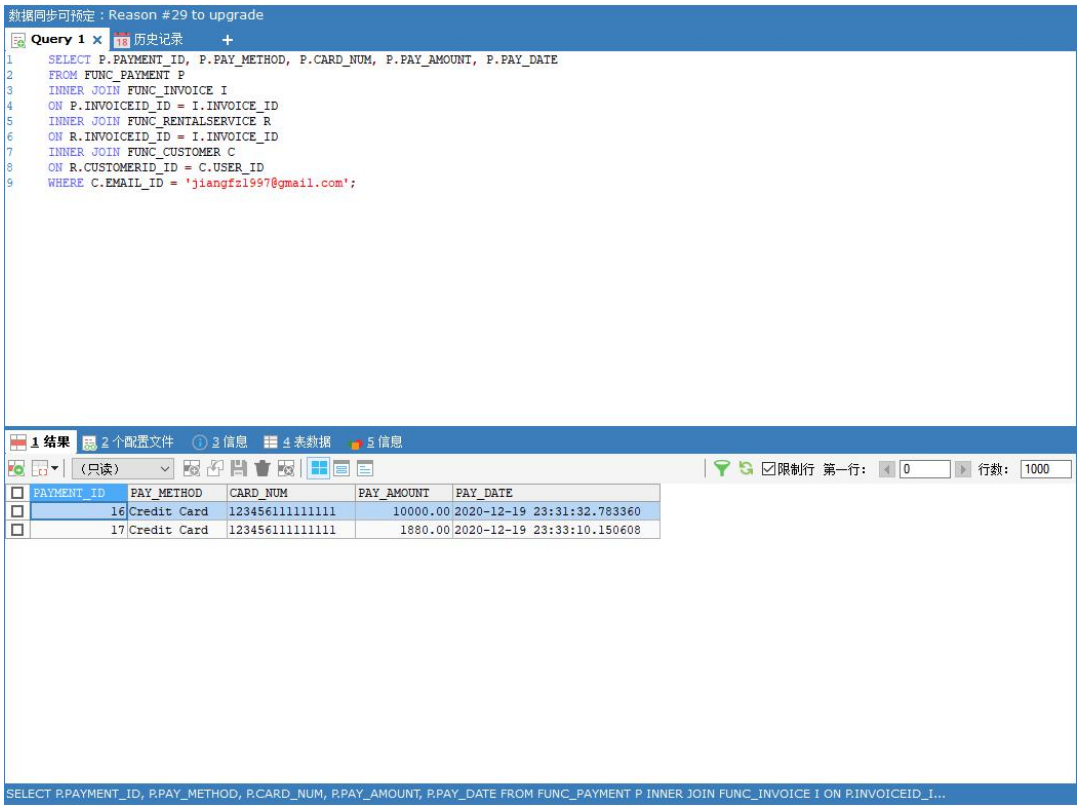
We have accomplished all the requirement and several extra feather, but there are still a lot of things we can do to improve our website, like using the distributed database, interface optimization, service expansion and many more. So I believe after this semester, we will keep working on our website because we know that this project is only about the score, but the knowledge we need for our future.

Appendix:

Sql query:

(1). Find all the payment made by the customer whose email is ‘jiangfz1997@gmail.com’

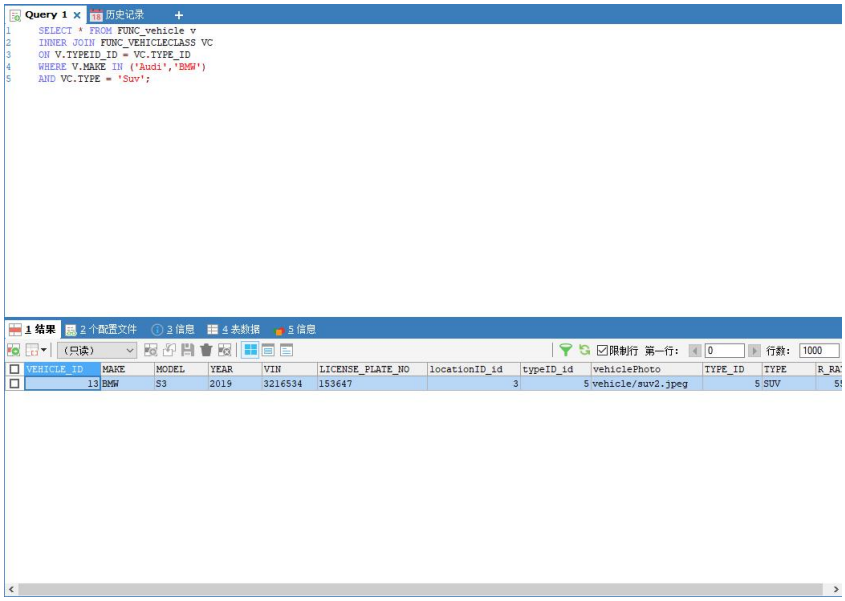
```
SELECT P.PAYMENT_ID, P.PAY_METHOD, P.CARD_NUM, P.PAY_AMOUNT, P.PAY_DATE

FROM FUNC_PAYMENT P

INNER JOIN FUNC_INVOICE I

ON P.INVOICEID_ID = I.INVOICE_ID

INNER JOIN FUNC_RENTALSERVICE R

ON R.INVOICEID_ID = I.INVOICE_ID

INNER JOIN FUNC_CUSTOMER C

ON R.CUSTOMERID_ID = C.USER_ID

WHERE C.EMAIL_ID = 'jiangfz1997@gmail.com';
```



(2).

Select all the SUV made by Audi or BMW.

```
SELECT * FROM FUNC_vehicle v

INNER JOIN FUNC_VEHICLECLASS VC

ON V.TYPEID_ID = VC.TYPE_ID

WHERE V.MAKE IN ('Audi','BMW')

AND VC.TYPE = 'Suv';
```



```
SELECT * FROM FUNC_vehicle v
INNER JOIN FUNC_VEHICLECLASS VC
ON V.TYPEID_ID = VC.TYPE_ID
WHERE V.MAKE IN ('Audi','BMW')
AND VC.TYPE = 'Suv';
```

VEHICLE_ID	MAKE	MODEL	YEAR	VIN	LICENSE_PLATE_NO	locationID_id	typeID_id	vehiclePhoto	TYPE_ID	TYPE	R_RA
13	BMW	S3	2019	3216534	153647	3	5	vehicle/suv2.jpeg	5	SUV	5

(3) .Select the top 3 who paid the most:

SELECT SUM(PAY_AMOUNT) AS TOTAL_PAYMOUNT, A.EMAIL_ID FROM

(SELECT P.PAYMENT_ID, P.PAY_METHOD, P.CARD_NUM, P.PAY_AMOUNT AS PAY_AMOUNT, P.PAY_DATE, C.EMAIL_ID AS EMAIL_ID

FROM FUNC_PAYMENT P

INNER JOIN FUNC_INVOICE I

ON P.INVOICEID_ID = I.INVOICE_ID

INNER JOIN FUNC_RENTALSERVICE R

ON R.INVOICEID_ID = I.INVOICE_ID

INNER JOIN FUNC_CUSTOMER C

ON R.CUSTOMERID_ID = C.USER_ID) AS A

GROUP BY A.EMAIL_ID

ORDER BY TOTAL_PAYMOUNT DESC

LIMIT 3;

```
SELECT SUM(PAY_AMOUNT) AS TOTAL_PAYMOUNT, A.EMAIL_ID FROM
(SELECT P.PAYMENT_ID, P.PAY_METHOD, P.CARD_NUM, P.PAY_AMOUNT AS PAY_AMOUNT, P.PAY_DATE, C.EMAIL_ID AS EMAIL_ID
FROM FUNC_PAYMENT P
INNER JOIN FUNC_INVOICE I
ON P.INVOICEID_ID = I.INVOICE_ID
INNER JOIN FUNC_RENTALSERVICE R
ON R.INVOICEID_ID = I.INVOICE_ID
INNER JOIN FUNC_CUSTOMER C
ON R.CUSTOMERID_ID = C.USER_ID) AS A
GROUP BY A.EMAIL_ID
ORDER BY TOTAL_PAYMOUNT DESC
LIMIT 3;
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

(只读) 限制行 第一行:

TOTAL_PAYMOUNT	EMAIL_ID
14400.00	i3@test.com
14000.00	indi2@test.com
11880.00	jiangfzl997@gmail.com

(4) . Calculate the average odometer of SUV for each rental service

WITH TEMP AS

(SELECT A.VEHICLE_ID, B.TYPE AS TYPE FROM FUNC_VEHICLE AS A INNER JOIN FUNC_VEHICLECLASS AS B ON A.TYPEID_ID = B.TYPE_ID)

SELECT AVG(R.END_ODOMETER - R.START_ODOMETER), T.TYPE

FROM FUNC_RENTALSERVICE R, TEMP T

GROUP BY T.TYPE

HAVING T.TYPE = 'SUV';

```sql
WITH TEMP AS
(SELECT A.VEHICLE_ID, B.TYPE AS TYPE FROM FUNC_VEHICLE AS A INNER JOIN FUNC_VEHICLECLASS AS B ON A.TYPEID_ID = B.TYPE_ID)
SELECT AVG(R.END_ODOMETER - R.START_ODOMETER), T.TYPE
FROM FUNC_RENTALSERVICE R, TEMP T
GROUP BY T.TYPE
HAVING T.TYPE = 'SUV';
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

(只读) 限制行 第一行: 0 行数: 1000

AVG(R.END_ODOMETER - R.START_ODOMETER)	TYPE
916.666667	SUV