

CS-GY 6083 B PRNCPLS DATABASE
SYSTEMS Section B final project part 2
report

Name: Siyuan Shi
Net ID: ss13376
Student ID: N18648680

Name: Haotian Yi
Net ID: hy1651
Student ID: N18800809

May 7, 2020

1 Development Environments

Programming language: python 3.6.8

Web framework: Django 3.0.4

Database: MySQL Workbench 8.0 CE

IDE: PyCharm

1.1 Django Logic

This section introduces the logic of the project.

- (1) The first thing to do is of course create a new django project.
- (2) Then we need to connect the database to the project. This include creating authentication user for the schema, django *inspectdb* command, etc. Within the django project itself, first of all, *models.py* must contains all the entities that database schema contains.
- (3) *settings.py* contains configuration information about database and apps created in the project. *urls.py* is the root URL configuration of the project.
- (4) *static* directory is used to provide our project wide static assets.
- (5) Inside *views.py*, we can define Python function or class that takes a Web request and returns a Web response.
- (6) *templates* directory contains all the templates the project needed. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

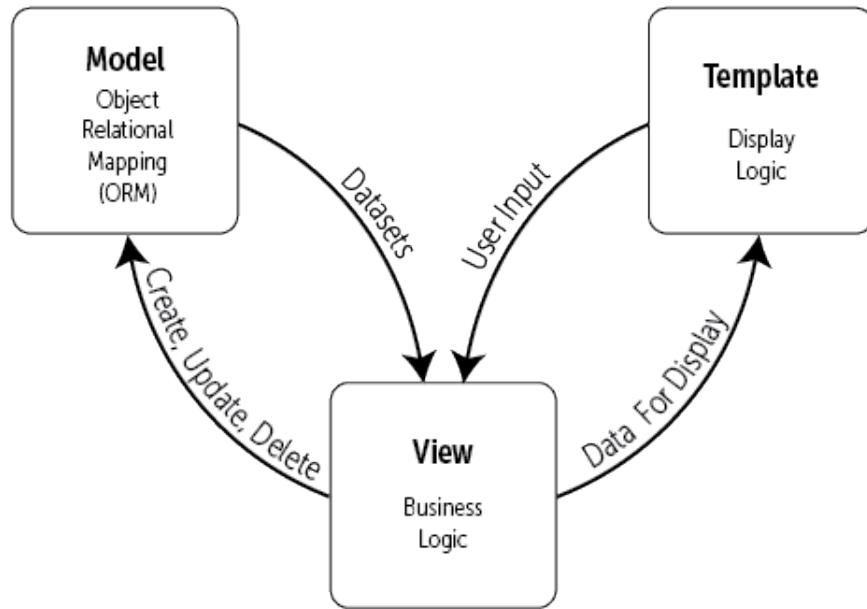


Figure 1: Django project logic

Figure 2 displays the running logic of django project. Model and View is on the server side, Template is on the client side. View can be used to create, update or delete records of model. It can also provide data for display. Template is used to display data for users. Users can input data via template.

2 Database

2.1 Relational Model

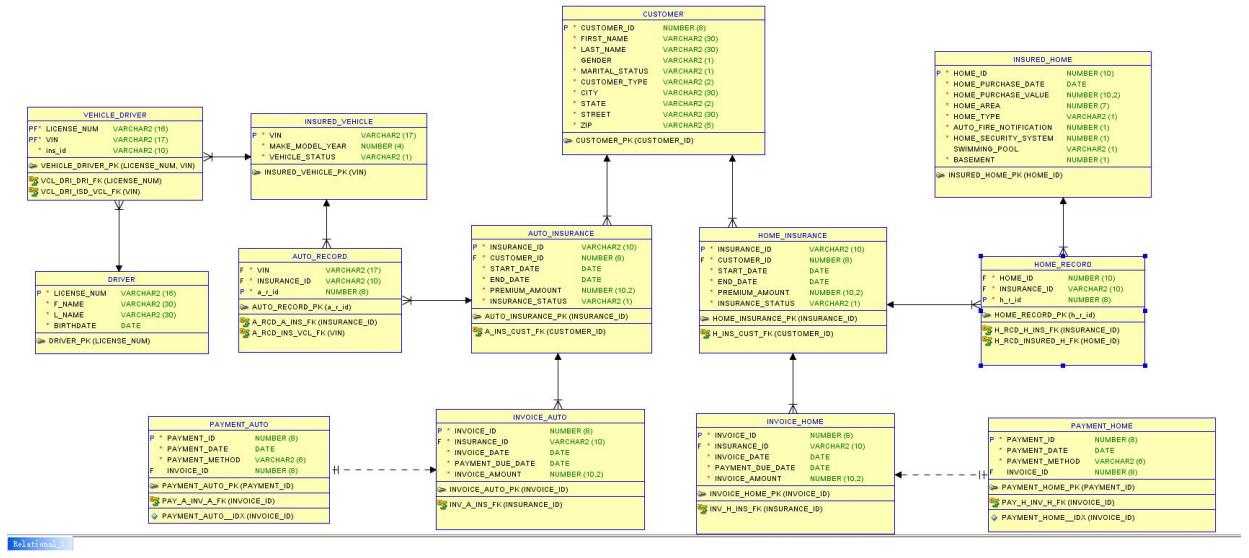


Figure 2: Relational Model

Full resolution picture of relational model is provided in a separate file.

2.2 DDL code

Here is part of DDL code:

```
1 -- Generated by Oracle SQL Developer Data Modeler 19.4.0.350.1424
2 -- at: 2020-05-05 15:46:41 EDT
3 -- site: Oracle Database 11g
4 -- type: Oracle Database 11g
5
6
7
8 CREATE TABLE auto_insurance (
9     insurance_id      VARCHAR2(10) NOT NULL,
10    customer_id        NUMBER(8) NOT NULL,
11    start_date         DATE NOT NULL,
12    end_date           DATE NOT NULL,
13    premium_amount     NUMBER(10, 2) NOT NULL,
14    insurance_status   VARCHAR2(1) NOT NULL
15 );
16
17 COMMENT ON COLUMN auto_insurance.insurance_id IS
18     'Insurance ID of auto-mobile insurance, starts with "A", followed by 9 digits number';
19
20 COMMENT ON COLUMN auto_insurance.customer_id IS
21     'Customer id';
22
23 COMMENT ON COLUMN auto_insurance.start_date IS
24     'start date of auto insurance policy';
25
26 COMMENT ON COLUMN auto_insurance.end_date IS
27     'end date of auto insurance policy';
28
29 COMMENT ON COLUMN auto_insurance.premium_amount IS
30     'auto insurance premium amount';
31
32 COMMENT ON COLUMN auto_insurance.insurance_status IS
33     'auto policy insurance status';
34
35 ALTER TABLE auto_insurance ADD CONSTRAINT auto_insurance_pk PRIMARY KEY ( insurance_id );
36
37 CREATE TABLE auto_record (
38     vin          VARCHAR2(17) NOT NULL,
39     insurance_id VARCHAR2(10) NOT NULL,
40     a_r_id       NUMBER(8) NOT NULL
41 );
42
43 COMMENT ON COLUMN auto_record.vin IS
```

Figure 3: DDL code (part)

Full code is provided in a separate sql file final.sql.

3 Features

This section depicts CRUD features and application procedure of our website.

3.1 Register, Login and Logout

This section introduces the sign up, login and logout feature of the project. Sign up view function will create a record in default User model with a primary key auto-created. During registering, make_password is called so that password is encrypted using the PBKDF2 algorithm (Password-Based Key Derivation Function 2) with a SHA256 hash. When logging in, the password typed will be encrypted and send to server. After logout, the session data for the current request is completely cleaned out.

The screenshot shows a login interface with the following elements:

- A title "Log in Your Account" centered at the top.
- A label "User name:" followed by a text input field.
- A label "Password:" followed by a text input field.
- A blue "Login" button at the bottom.

Figure 4: Login

Sign Up Your Account

User name:

Password:

Confirm password:

sign up

Figure 5: Sign up

3.2 Update | Personal Center

This section introduces the update feature of the project. A user can update personal information displayed in personal center. Insurance related information can also be modified, however, this operation is reserved for administrator.



Personal Information

Customer ID: 2 First Name: NATASHA Last Name: ROMANOFF

Gender: F Marital Status: W Customer Type: H

City: LOS ANGELES State: CN Street: 120 WESTWOOD Zip: 13501

Update

My Insurance

My Invoice

Figure 6: Personal Center

Update

First Name: NATASHA

Last Name: ROMANOFF

Gender: ▾

Marital Status: W

M-Married, S-Single, W-Widow/Widower

Customer Type: H

A-Auto Insurance, H-Home Insurance, B-Both. This is updated my system automatically.

City: LOS ANGELES

State: CN

Please input two letter abbreviation

Street: 120 WESTWOOD

Zip: 13501

Update

Figure 7: Update Info

3.3 Insert | Purchase

This section introduces the insert feature of the project. After user purchase an insurance, necessary records will be inserted into different tables of the database. Take home insurance for example:

- (1) Start page for home insurance:



Home Insurance:

Putting Down Roots

Protect your largest investment from unexpected events life may throw your way.



[Enroll](#)

Figure 8: Start page

(2) Fill insurance information:

Please fill up basic information

Start Date:

End Date:

We strongly recommend you to...

Premium Amount: US dollars

Premium amount must be between \$0.00 and \$99,999,999.99

[Next Step](#)

Your house has been insured before? [Purchase Again](#)

This is a WDS official website

Feel not so insured? [Get insured now! We do secure!](#).

Reminder: please click Account at the top to register information before enrolling for insurance.

Figure 9: Home Insurance

(3) Start date cannot be earlier than today:

Start Date: ▼

End Date:

We strongly recommend you to get insured at least for 1 year

Premium Amount:

Premium amount equals amount we insured for each month

Next Step

Your house has been insured before? Purchase Again

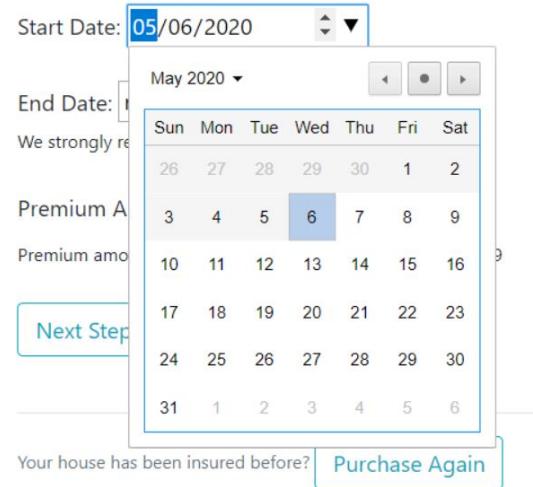


Figure 10: Start date restriction

(4) End date chosen by option:

Start Date:

Insured Year: months

We strongly recommend you to get insured at least for 1 year

Premium Amount: US dollars

Premium amount equals amount we insured for each month

Next Step

Your house has been insured before? Purchase Again

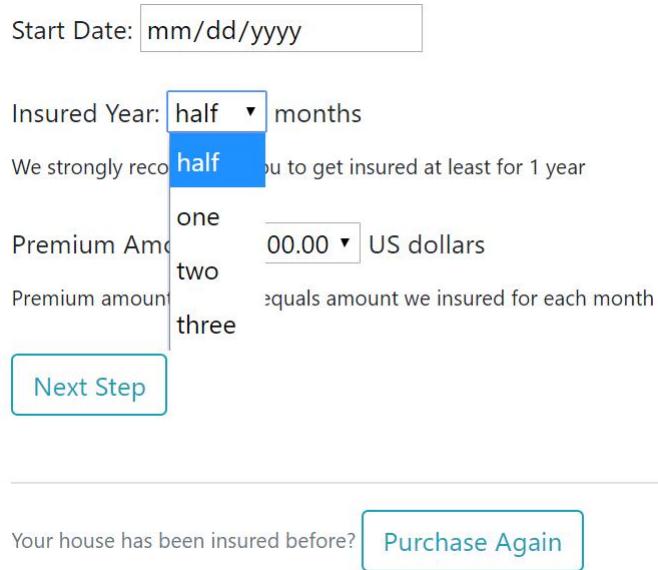


Figure 11: End date restriction

(5) Fill up insured home information:

The screenshot shows a form titled 'INSURE ME' with various input fields and dropdown menus. The fields include:

- Home ID: The unique identifier of your deed
- Home Purchase Date: When did you purchase your house?
- Price: US dollars The price of your house
- Home Area: Square Feet
- Home Type:
- Does your house have auto fire notification?
- Does your house have home security system?
- Type of Swimming Pool:
- Does your house have basement?
-

Figure 12: Insured Home

(6) View order information and pay:

The screenshot shows an 'Order Info' page with the following details:

Home ID: 12345
Insurance ID: 8720278443
Premium Amount: 2000.00 US dollars

Payment Options:

-
-

Figure 13: Order Info

(7) View invoice:

Invoice

Insurance Number: 8720278443
Invoice Number: 27427140
Invoice Date: May 6, 2020
Payment Due Date: May 21, 2020
Invoice Amount: 2000.00 US dollars

[Pay This Invoice](#)

Figure 14: Invoice

(8) Make payment:

Payment

Invoice Number: 27427140
Invoice Amount: 2000.00 US dollar
Payment Number: 58522562
Payment Date: May 6, 2020
Payment Method: Check ▾
Account Number :
[Checkout](#)

Figure 15: Payment

3.4 Query | Insurance,Invoice & Payment

This section introduces the query feature of the project. There are two kinds of query in this project, insurance query and invoice query. Insurance query consists of home insurance query and auto insurance query. User can only query on his or her own data, including insurance information, house information, vehicle information, driver information, etc. Invoice query is used for displaying user's invoices. User can also use this entrance to pay installment invoices.

(1) Query Insurance information:

We Do Secure

Update Insurance Status Delete Insurance Account Enroll Log out

Query Home Insurance

Here are insurance ID of yours

1100000021
8720278443

Please input insurance ID you want to search

Figure 16: Home Insurance Query

Insurance Info

Insurance ID: 8720278443

Start Date: May 6, 2020, midnight

End Date: Nov. 25, 2020, midnight

Premium Amount: 2000.00

Insurance_status: C

Figure 17: Home Insurance Info

Insured Home Info

Home ID: 12345
Home Purchase Date: Oct. 1, 2019, midnight
Price: 20000.00
Area: 2000
Type: S
Auto Fire Notification: 1
Home Security System: 1
Swimming Pool: M
Basement: 1

[Query More](#)

Figure 18: Home Info

(2) Query invoice and payment: this is designed for users to check all their invoice details and pay installment. If one choose to pay installments, there will be multiple invoice for one insurance.

The screenshot shows a web page titled "Query Home Insurance Invoice". At the top, there is a navigation bar with links for "Update Insurance Status", "Delete Insurance", "Account", "Enroll", and "Log out". Below the title, a sub-header reads "Here are insurance ID of yours". A large button labeled "Detail" is centered. At the bottom of the page, there is footer text: "This is a WDS official website", "Feel not so insured? Get insured now! We do secure!", "Reminder: please click Account at the top to register information before enrolling for insurance.", and a link "Back to top".

Figure 19: Home Invoice Query

The screenshot shows a web page titled "Invoice list". At the top right, there are links for "Update Insurance Status", "Delete Insurance", "Account", "Enroll", and "Log out". On the left, there is a logo for "We Do Secure". Below the title, a blue banner reads: "Please keep checking and paying your invoice in your "Account->My Invoice"". Underneath the banner, there is a summary of the invoice: "Invoice Number: 27427140", "Invoice Date: May 6, 2020, midnight", and "Status: Paid". A button labeled "Pay This Invoice" is present, with a note below it stating "You already paid this invoice!". At the bottom of the page, there are footer links: "This is a WDS official website", "Feel not so insured? Get insured now! We do secure!", "Reminder: please click Account at the top to register information before enrolling for insurance.", and "Back to top".

Figure 20: Home Invoice Query Result

3.5 Delete | Superuser Function

This section introduces the delete feature of the project. Delete feature is designed for administrator. An administrator can delete a certain insurance record and all related information.

(1) Input insurance ID to delete:

The screenshot shows a web page titled "Delete". At the top right, there are links for "Update Insurance Status", "Delete Insurance", "Account", "Enroll", and "Log out". On the left, there is a logo for "We Do Secure". Below the title, there is a search bar containing the insurance ID "8720278443" and a "Search" button. At the bottom of the page, there are footer links: "This is a WDS official website", "Feel not so insured? Get insured now! We do secure!", "Reminder: please click Account at the top to register information before enrolling for insurance.", and "Back to top".

Figure 21: Delete

(2) Confirm delete:

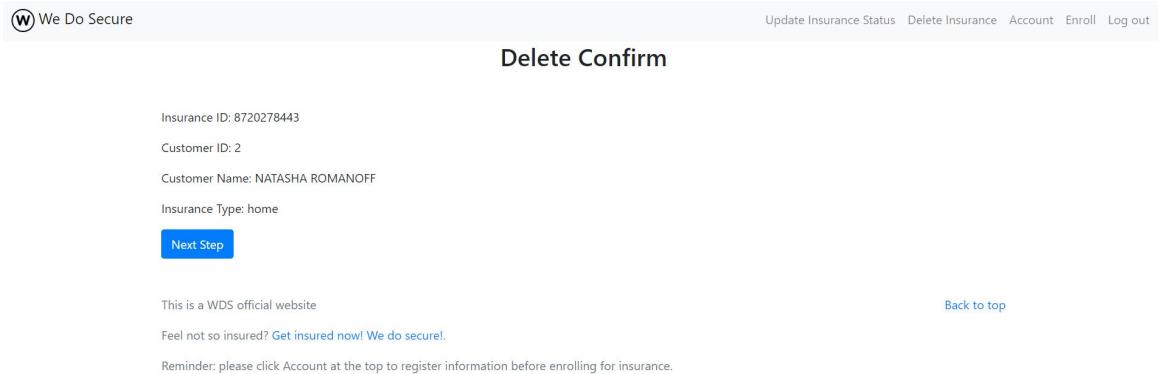


Figure 22: Delete Confirm

(3) Click to delete and lift result message:

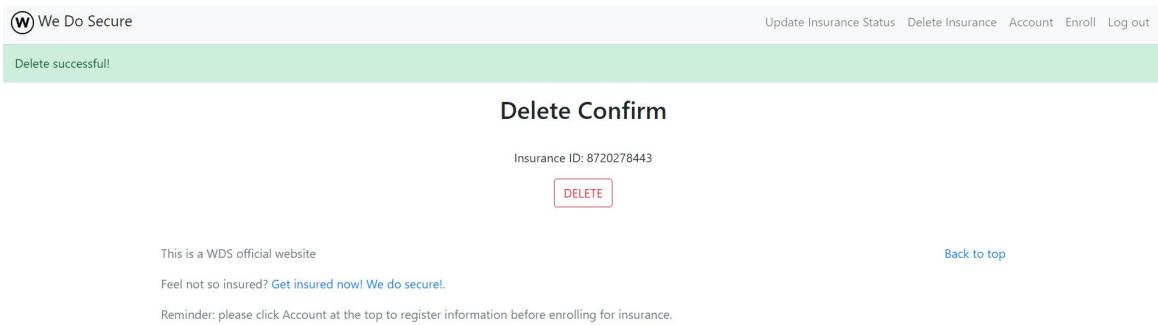


Figure 23: Delete Result

3.6 Extra Feature Summary

(1) Index

```
class AutoInsurance(models.Model):
    insurance_id = models.CharField(db_index=True, primary_key=True, max_length=10)
```

Figure 24: Index

For the index feature, we follow the thumb rule that we build index on attributes that are frequently queried. In Django, it is easy to enable index feature, with the statement `db_index=True` above, it builds a B+ tree index on the attribute.

(2) Password encryption

```
except User.DoesNotExist:
    if password1 == password2:
        user = User.objects.create_user(username=username, password=password1)
        user.password = make_password(password1) # encryption
        print('Encoded password:',make_password(password1))
        print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
        user.save()
```

Figure 25: Encryption

After create user, change password attribute by using `make_password()` to encrypt. Then use `auth.authenticate()` below to authenticate users to login.

```
user = auth.authenticate(username=username, password=password)
```

Figure 26: Authenticate

(3) CSRF

```
<form method="post" action="{% url 'auto_ins' %}>
    {% csrf_token %}
```

Figure 27: CSRF

Add `{% csrf_tocken %}` to get protection from CSRF (Cross Site Request Forgery) attack.

(4) Status Update (automatic)

```
def update_ins_status(request):
    #-----update insurance status-----
    TODAY = datetime.datetime.utcnow()
    auto = models.AutoInsurance.objects.all()
    for a in auto:
        if TODAY > a.end_date.replace(tzinfo=None):
            models.AutoInsurance.objects.filter(insurance_id=a.insurance_id).update(insurance_status='P')
    home = models.HomeInsurance.objects.all()
    for h in home:
        if TODAY > h.end_date.replace(tzinfo=None):
            models.HomeInsurance.objects.filter(insurance_id=h.insurance_id).update(insurance_status='P')
    #-----update customer type-----
    customer = models.Customer.objects.all()
    for c in customer:
        a = models.AutoInsurance.objects.filter(customer_id = c.customer_id,insurance_status='C')
        h = models.HomeInsurance.objects.filter(customer_id = c.customer_id,insurance_status='C')
        if a.exists() and h.exist():
            models.Customer.objects.filter(customer_id = c.customer_id).update(customer_type = 'B')
        elif a.exists():
            models.Customer.objects.filter(customer_id=c.customer_id).update(customer_type='A')
        elif h.exists():
            models.Customer.objects.filter(customer_id=c.customer_id).update(customer_type='H')
    return render(request, 'update_ins_status.html')
```

Figure 28: Status Update

Achieve a function for superuser to update insurance status and customer type, also, this update function is embedded in personal center (interface to view related information), every time a user access personal center, it will automatically update status of current user.

(5) SQL Injection

```
home_ins = models.HomeInsurance.objects.filter(customer_id=customer_id)
```

Figure 29: SQL Injection

In the aspect of SQL Injection, we prevent this by using Django ORM function, it process input by only accepting value for corresponding attribute, thus protect from injection.

(6) jQuery

```
<script>
    var dtToday = new Date();

    var month = dtToday.getMonth() + 1;
    var day = dtToday.getDate();
    var year = dtToday.getFullYear();
    if(month < 10)
        month = '0' + month.toString();
    if(day < 10)
        day = '0' + day.toString();

    var minDate = year + '-' + month + '-' + day;
    $('#s').attr('min', minDate);
</script>

<script>
    $("#s").change(function(){
        $('#e').attr('min', $('#s').val());
    });
</script>
```

Figure 30: jQuery

When designing template for home insurance and auto insurance, we use jQuery to dynamically set attribute for certain input tag (start date) so that user cannot choose a date before today as the start date for home or auto insurance. We also use option to restrict end date such that user cannot choose a date earlier than start date. These restrictions for input is essential for the stable of database.

(7) Beautified UI

Figures below are UI sample we beautified by HTML, CSS, bootstrap templates.

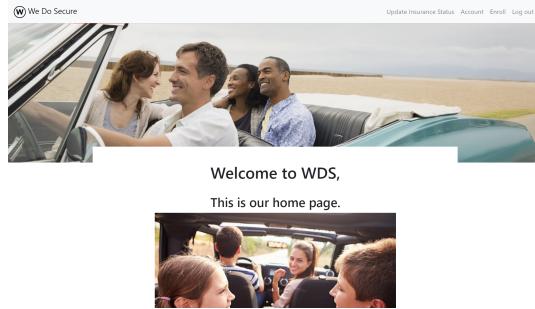


Figure 31: UI sample -1

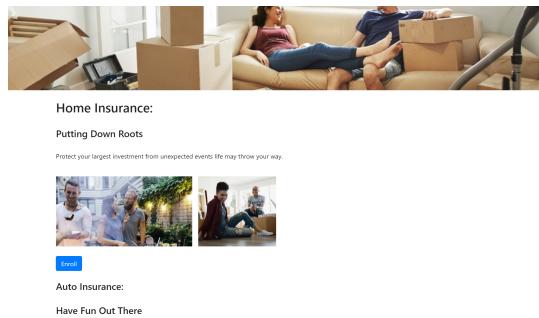


Figure 32: UI sample -2

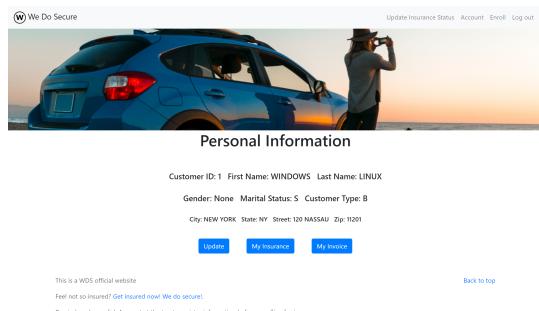


Figure 33: UI sample -3

4 Learning Outcome

(1) About input constraint

When considering about input for a practical website, we have to think about constraints, that doesn't mean we let invalid input trigger error in database, we should validate input at web page and lift reminder once a input is invalid.

Also we have to consider correlation between inputs like start date and end date, so we used jquery to limit selection of date.

(2) About passing argument

In this project, we used different ways to passing argument. Argument can be passed via render or redirect function. We can also use *href* attribute in html file to achieve redirecting with parameters. Another way is to store variables using *session* such that other function can use the stored variables in the same session.

(3) About procedure of design

After this project, we are more familiar with the procedure of building a website and its application, especially we feel that we must be careful when design a conceptual model for business and database because a good design can really save time in building website page and alteration for logic mistakes. Moreover, we think there will always be problems we can't predict when designing database models in advance but encounter in practice.