The part I am responsible for is Airline, which is a system relatively close to the management level. Because three people in our group are responsible for this area, we also need to have some differences. Therefore, I have implemented a backend management system, including the Airline section, and added some new functions, as well as partially verifying data and requests. It includes the Airline section and has also made more extensions, so don't be surprised to see the new sections added. This article only discusses the project framework, the composition and corresponding functions of each part, and the database structure. The complete website application is displayed in the guide.

https://github.com/frankRenlf/Web_Services_Data/blob/main/cw2/README.md

The whole project is designed by django, and some requirements and corresponding versions are list there:

asgiref==3.6.0

Django==4.1.7

djangorestframework==3.14.0

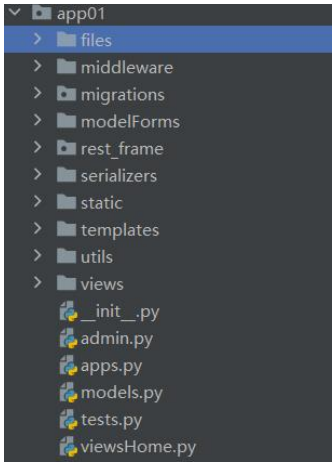
mysqlclient==2.1.1

Pillow==9.4.0

pytz==2023.3

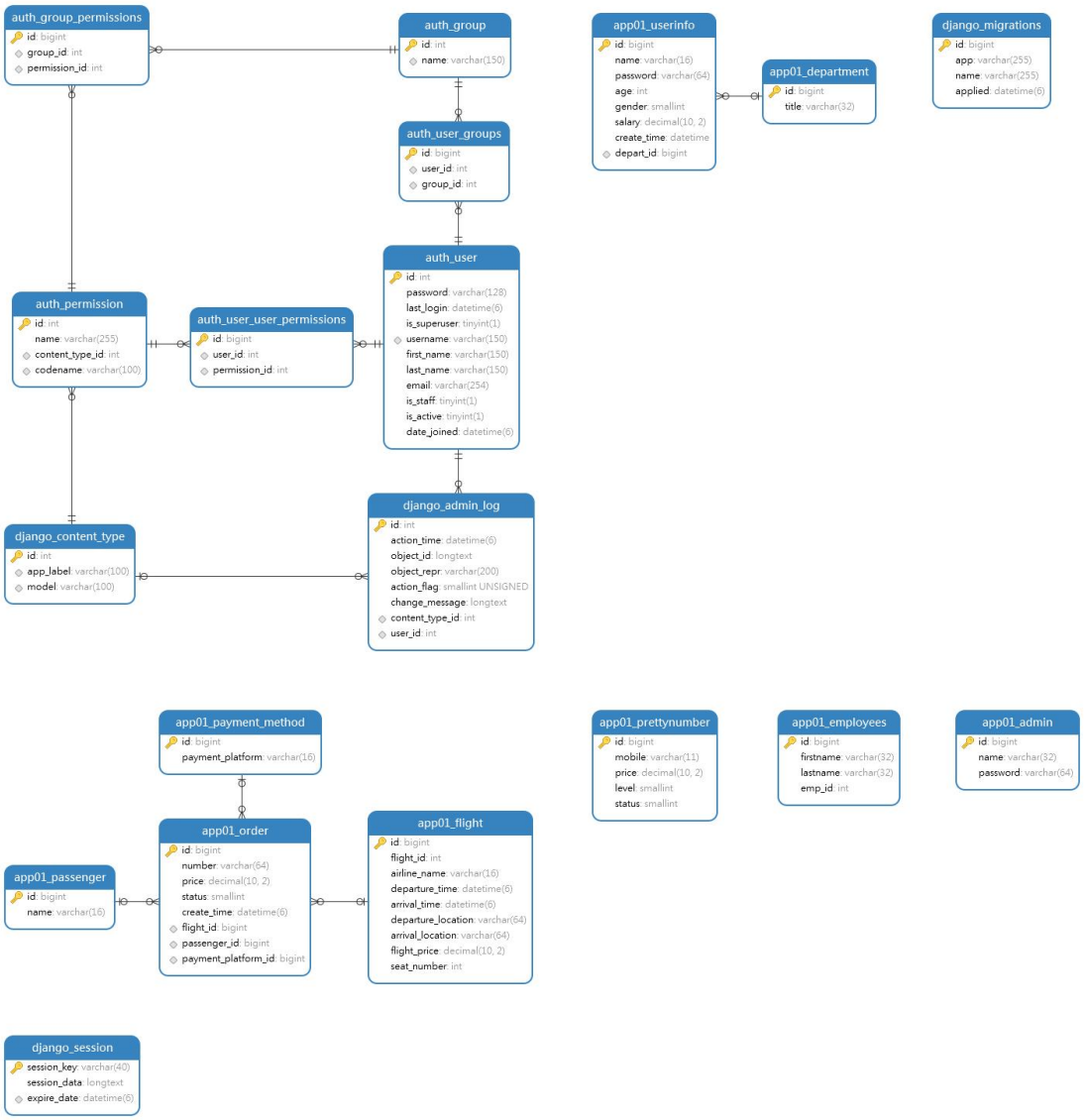
sqlparse==0.4.4

requests==2.30.0



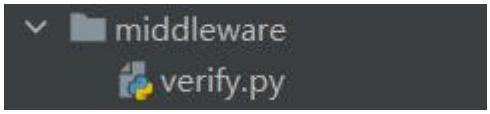
The generated application has the following structure:

In this project, I choose mysql as the database for it's fast selection and stable usage. And this is the structure of part of the project.

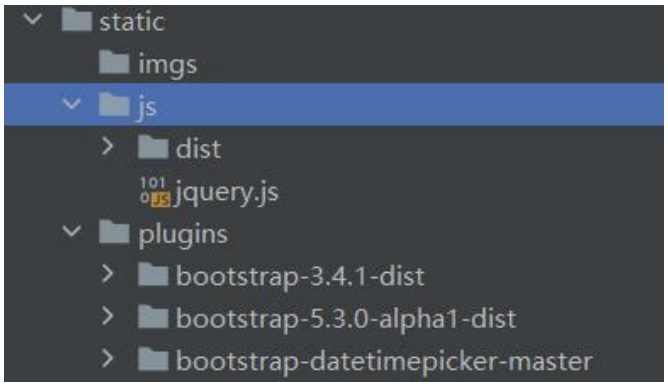


The next part I will introduce the detail in project structure.

The files dictionary is used to contain the upload files, with # upload path('upload/list', upload.upload_list), path('upload/form', upload.upload_form). It is one additional method.

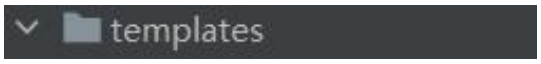


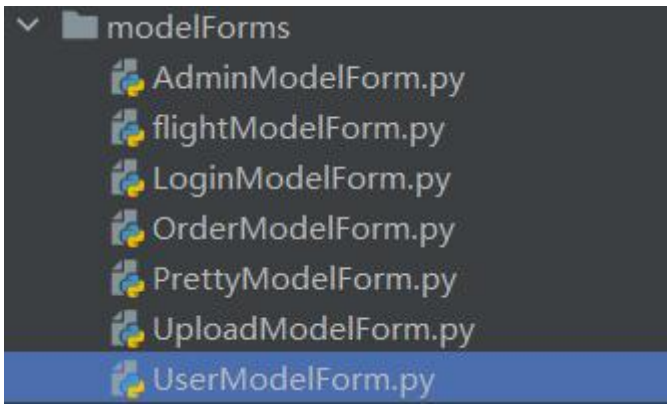
This module is used to build middleware, to intercept the request and determine whether it has passed information verification. If it passes, release it, otherwise it will redirect to the login interface. At the same time, use session for persistence verification session.set_ Expiry (60), information needs to be revalidated after expiration.



This module is used to provide some CSS and JavaScript support, making the interface more beautiful and reasonable. At the same time, jquery can also make requests more annotated, in line with restful style.

Whole pages are listed there.





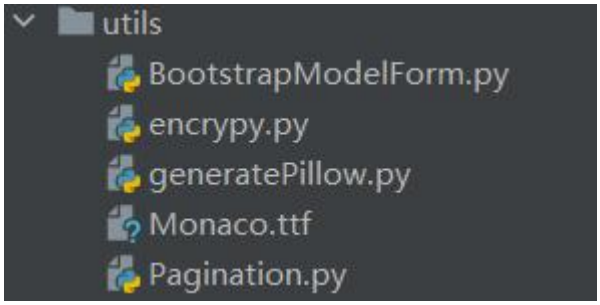
These are some modelform classes that I use to encapsulate the basic models class while adding more functionality. Some advanced technologies are used.

```
from app01.utils.BootstrapModelForm import BootstrapModelForm
from app01 import models
from django import forms
from django.core.exceptions import ValidationError
from app01.utils.encrypy import md5
```

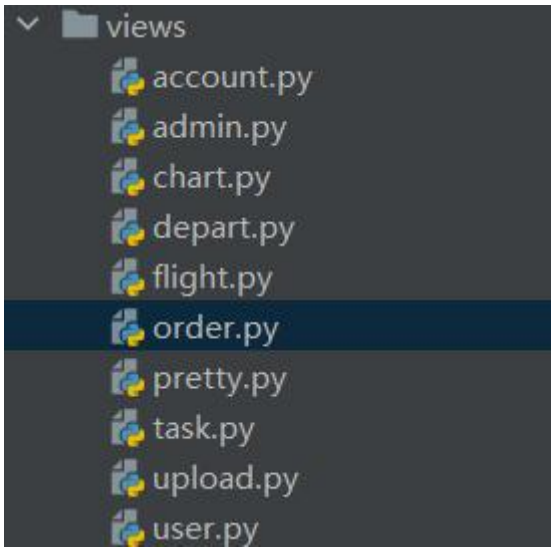
These parts are used for functional extensions, such as BootstrapModelForm,

Forms, etc. There are also some such as ValidationError, md5, which are used for testing, error verification, and encryption.

There exists one file called BootstrapModelForm from utils dictionary, and it will be esplained later.



This module is used by me to set the base class BootstrapModelForm for style classes, encrypt encryption, and encapsulate pagination for pagination and search functions, and this part really works. At the same time, generatePillow also uses the Monaco.ttf file to set the font and layer of the verification code, also for security reasons, I use a graphical verification code. If verification is required, it is best to log in through the web. Otherwise, using Postman and converting to web browsing is also possible.



The last part is for the views dictionary which is used to output the corresponding functions of each URL, such as interface, JSON data, etc.

There are abundant url, so I will not list there, just go to the web page, follow the instructions and test them.

The interaction with other subsystems is completed within the system by

requests module, and if testing is required, I have also exposed some interfaces for inspection, which is convenient to use. If you want to see specific data accessed from other systems on the interface, you can use the following interface, some require permissions and can be processed internally within the system, but individual access may be restricted:

http://sc19lr.pythonanywhere.com/payment/?api=invoice&aid=1&oid=2&ta=22&al=Frank&dm=200

http://sc19lr.pythonanywhere.com/payment/?api=pay&aid=1&oid=2&ta=22&al=Frank&dm=200

Each param can be changed except al=Frank, use different variables to test the different apis. However, there may some permission issues, in order to prevent malicious access from others, some security measures will be taken for exposed interfaces.

If you want to see specific data, you can use the following interface, just for instance:

http://sc19lr.pythonanywhere.com/order_data/

http://sc19lr.pythonanywhere.com/flight_data/

As for the test part, several api tests are done by apifox

As for the test unit part in the project, many tests are prepared, youcan find in app01\tests.py, and you can run the command: python manage.py test app01,

```
(venv) PS C:\Users\11195\OneDrive - University of Leeds\Desktop\year4\Web Services and Web Data
Found 15 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.........{}
.....
----------------------------------------------------------------------
Ran 15 tests in 0.209s

OK
Destroying test database for alias 'default'...
```