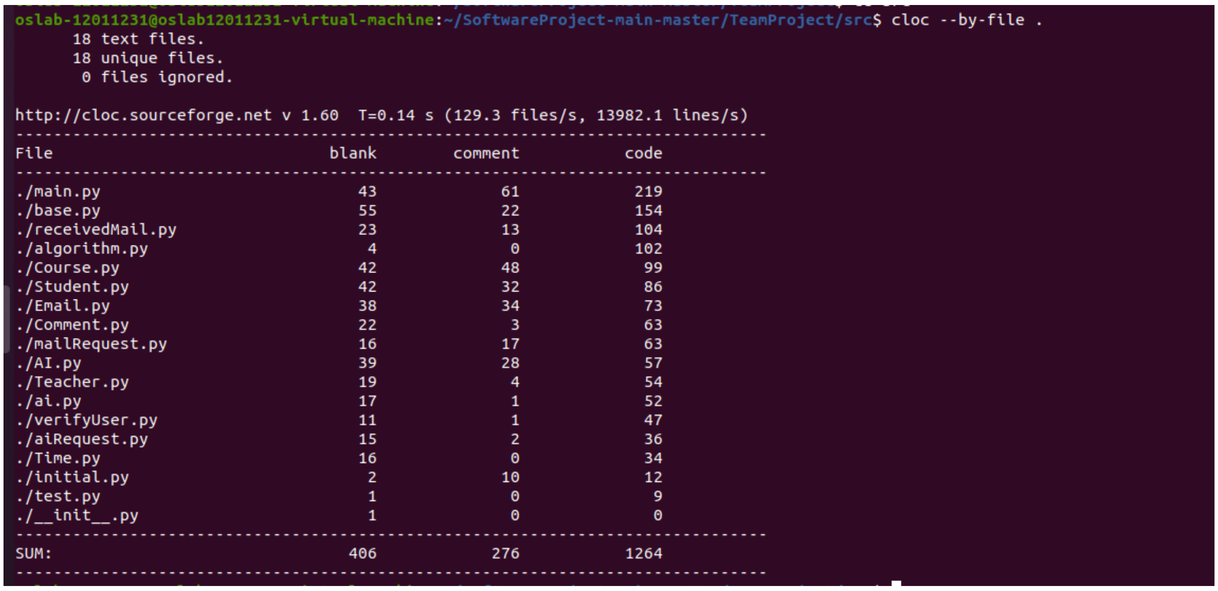
Documentation for developers

1. Metrics

For back-end, we use cloc to count the lines of Code and number of source files:

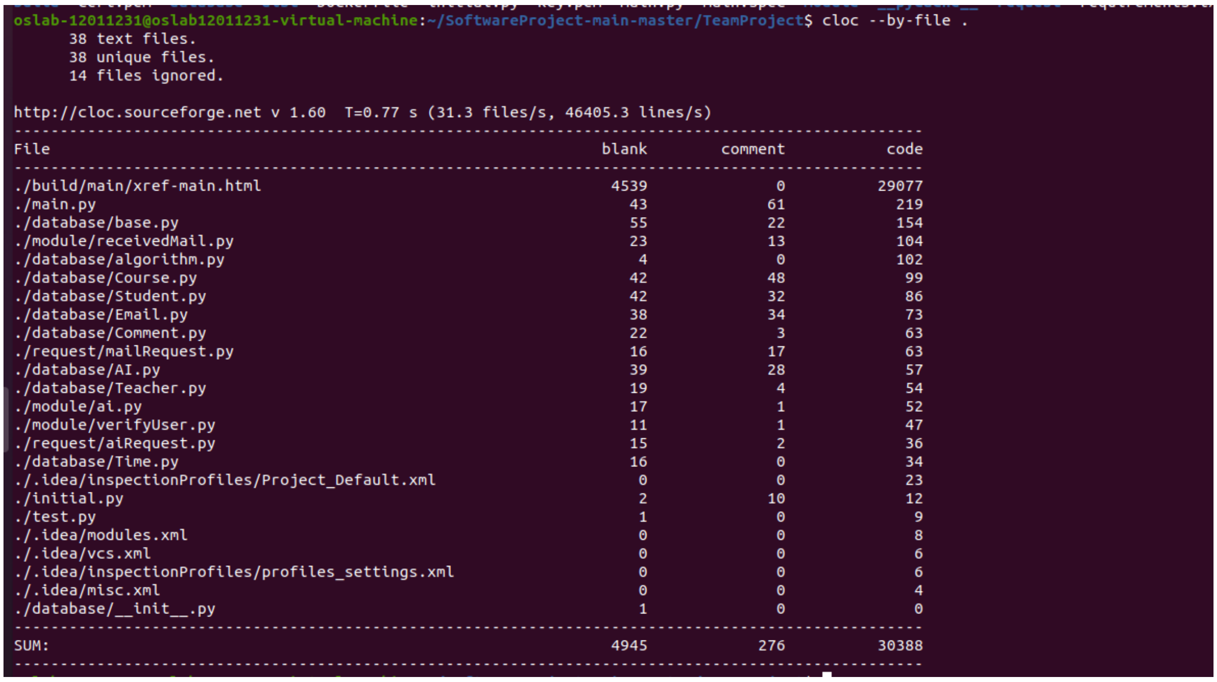
For src:



```
oslab-12011231@oslab12011231-virtual-machine:~/SoftwareProject-main-master/TeamProject/src$ cloc --by-file .
      18 text files.
      18 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.60  T=0.14 s (129.3 files/s, 13982.1 lines/s)
-------------------------------------------------------------------------------
File                              blank        comment           code
-------------------------------------------------------------------------------
./main.py                            43             61            219
./base.py                            55             22            154
./receivedMail.py                    23             13            104
./algorithm.py                        4              0            102
./Course.py                          42             48             99
./Student.py                         42             32             86
./Email.py                           38             34             73
./Comment.py                         22              3             63
./mailRequest.py                     16             17             63
./AI.py                              39             28             57
./Teacher.py                         19              4             54
./ai.py                              17              1             52
./verifyUser.py                      11              1             47
./aiRequest.py                       15              2             36
./Time.py                            16              0             34
./initial.py                          2             10             12
./test.py                             1              0              9
./__init__.py                         1              0              0
-------------------------------------------------------------------------------
SUM:                                406            276           1264
-------------------------------------------------------------------------------
```

For whole back-end project:



```
oslab-12011231@oslab12011231-virtual-machine:~/SoftwareProject-main-master/TeamProject$ cloc --by-file .
      38 text files.
      38 unique files.
      14 files ignored.

http://cloc.sourceforge.net v 1.60  T=0.77 s (31.3 files/s, 46405.3 lines/s)
-----------------------------------------------------------------------------------------------
File                                                 blank        comment           code
-----------------------------------------------------------------------------------------------
./build/main/xref-main.html                           4539              0          29077
./main.py                                               43             61            219
./database/base.py                                      55             22            154
./module/receivedMail.py                                23             13            104
./database/algorithm.py                                  4              0            102
./database/Course.py                                    42             48             99
./database/Student.py                                   42             32             86
./database/Email.py                                     38             34             73
./database/Comment.py                                   22              3             63
./request/mailRequest.py                                16             17             63
./database/AI.py                                        39             28             57
./database/Teacher.py                                   19              4             54
./module/ai.py                                          17              1             52
./module/verifyUser.py                                  11              1             47
./request/aiRequest.py                                  15              2             36
./database/Time.py                                      16              0             34
./.idea/inspectionProfiles/Project_Default.xml           0              0             23
./initial.py                                             2             10             12
./test.py                                                1              0              9
./.idea/modules.xml                                      0              0              8
./.idea/vcs.xml                                          0              0              6
./.idea/inspectionProfiles/profiles_settings.xml         0              0              6
./.idea/misc.xml                                         0              0              4
./database/__init__.py                                   1              0              0
-----------------------------------------------------------------------------------------------
SUM:                                                  4945            276          30388
-----------------------------------------------------------------------------------------------
```

Lines of Code: 1264

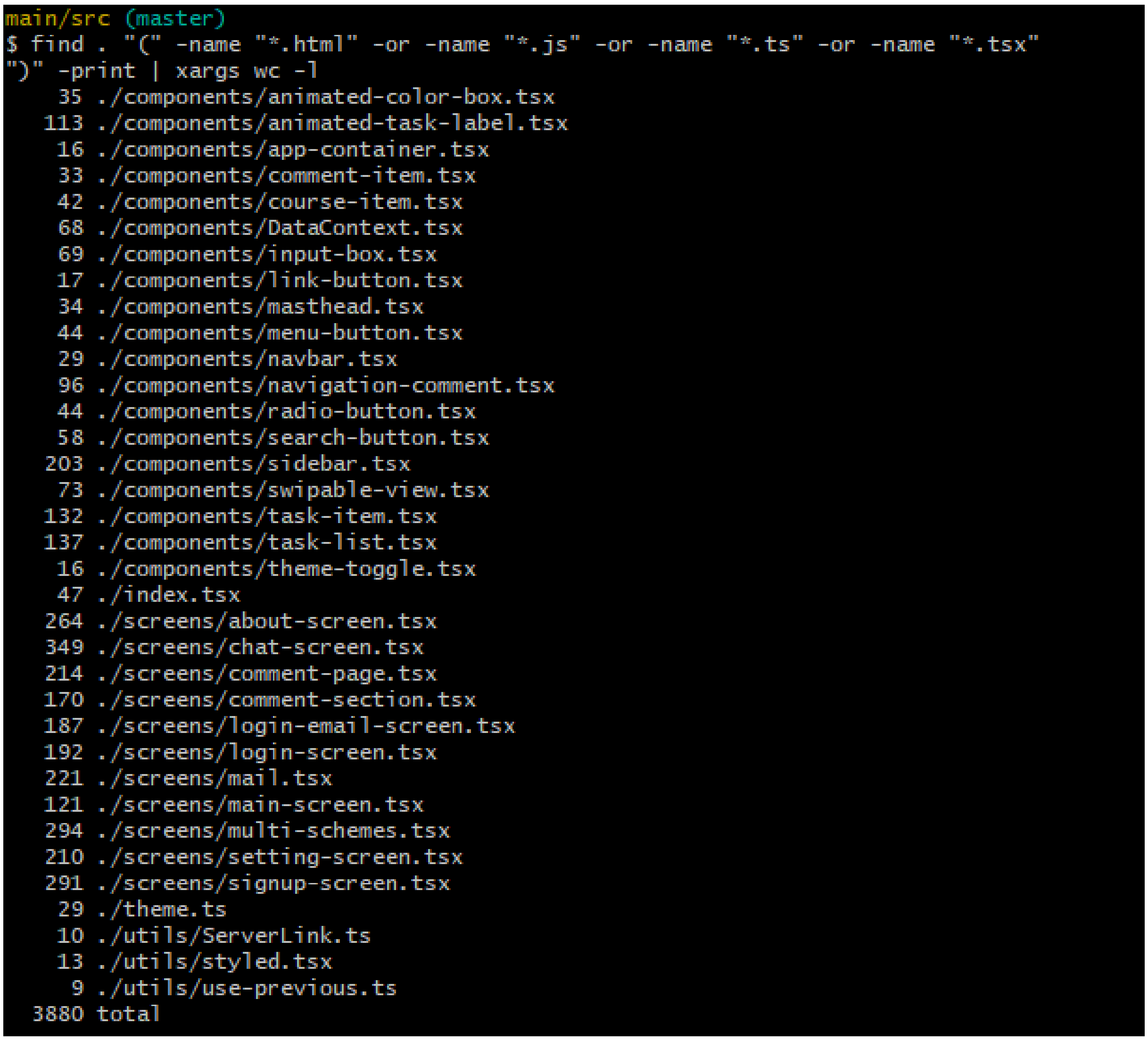
Number of modules: 18

At the same time, we can get the number of source files and number of packages.

Number of packages: 3

Number of source files: 38

For front-end, we use git bash to count the lines of Code.



```
main/src (master)
$ find . "(" -name "*.html" -or -name "*.js" -or -name "*.ts" -or -name "*.tsx"
")" -print | xargs wc -l
    35 ./components/animated-color-box.tsx
   113 ./components/animated-task-label.tsx
    16 ./components/app-container.tsx
    33 ./components/comment-item.tsx
    42 ./components/course-item.tsx
    68 ./components/DataContext.tsx
    69 ./components/input-box.tsx
    17 ./components/link-button.tsx
    34 ./components/masthead.tsx
    44 ./components/menu-button.tsx
    29 ./components/navbar.tsx
    96 ./components/navigation-comment.tsx
    44 ./components/radio-button.tsx
    58 ./components/search-button.tsx
   203 ./components/sidebar.tsx
    73 ./components/swipable-view.tsx
   132 ./components/task-item.tsx
   137 ./components/task-list.tsx
    16 ./components/theme-toggle.tsx
    47 ./index.tsx
   264 ./screens/about-screen.tsx
   349 ./screens/chat-screen.tsx
   214 ./screens/comment-page.tsx
   170 ./screens/comment-section.tsx
   187 ./screens/login-email-screen.tsx
   192 ./screens/login-screen.tsx
   221 ./screens/mail.tsx
   121 ./screens/main-screen.tsx
   294 ./screens/multi-schemes.tsx
   210 ./screens/setting-screen.tsx
   291 ./screens/signup-screen.tsx
    29 ./theme.ts
    10 ./utils/ServerLink.ts
    13 ./utils/styled.tsx
     9 ./utils/use-previous.ts
  3880 total
```

Lines of Code: 3880

Number of source files: 39

Number of modules: 35

Number of packages: 5

Front-end

1. 引言

The front-end part of this project provides users with an interface to interact with various functionalities of the back-end. It offers users a mobile visual interactive interface for scheduling, course program generation, course comment section, email summaries and ratings, and an AI assistant.

2. 安装和配置

packages	version
react	18.0.0
react-dom	18.0.0
react-native	0.69.9
react-native-sage-area-context	4.3.1
react-native-svg	12.3.0
npm	9.5.0
nodejs	18.15.0
yarn	1.22.19
expo go (iOS or Android)	2.28.9

3. 快速入门

Here is a useful website for beginners to learn React-Native: https://reactnative.cn/

Clone the front-end project fromhttps://github.com/YuuKiriyama/final_front_end

Set up the environment of react-native, nodejs, npm and yarn.

Download Expo Go from App store or Google Play on your mobile device.

Run command `yarn start`.

Make sure the mobile device and your server under the same subnet.

Scan the QR code on the terminal with Expo Go, then you can see the front-end preview.

4. 核心功能和实现

Login & Register

- Function Description: Signup with your email address to get access in the app. You can login with your id and password or simply use Email verification code.
- Implementation Method Code Location:
  src/screen/login-screen.tsx
  src/screen/signup-screen.tsx
  src/screen/login-email-screen.tsx
  src/components/DataContext.tsx
- Important Notes: Only the latest email with verification code counts.

Tasks

- Function Description: Add, edit and delete task enties to manage your task list.
- Implementation Method Code Location:
  /src/screens/main-screen.tsx
  /src/components/task-item.tsx
  /src/components/task-list.tsx
- Important Notes: The deleting process will fail in ipad due to screen size.

Course Scheduling

- Function Description: Choose the courses, and you will be given a list of features describing all of the plans generated. Then you can go into each plan to see details.
- Implementation Method Code Location: /src/screens/about-screen.tsx
- Important Notes: The table of the course plans is designed and adjusted on iPhone 12 pro. It may have some problems on other untested devices.

Comment

- Function Description: Search to find the course interests you and check the previous comments left by others.
- Implementation Method Code Location:
  /src/screens/comment-section.tsx

/src/screens/comment-page.tsx

/src/components/comment-item.tsx

/src/components/course-item.tsx

- Important Notes: If you are trying to get a large list, the app may not render in time.

Mail simplification

- Function Description: After your registration, you can click requery to find the abstract of your latest mail and its important level estimated according to your self introduction.
- Implementation Method Code Location: /src/screens/mail.tsx
- Important Notes:Because everyone's mailboxes have different contents, the app may not display your latest mail correctly under some special circumstances. Such as an empty mail or a mail without text.

AI assistant

- Function Description: You can talk to the AI assistant and get some advice from the AI.
- Implementation Method Code Location: /src/screens/chat-screen.tsx
- Important Notes: If the answer is a bit long, you may need to wait for a while to get the response.

Back-end

1. 引言

Our project aims to develop a platform for college students that enhances efficiency in handling tasks and provides information about campus life. Students can use our platform to manage their schedules, access course reviews, automatically generate class schedules based on their selected courses, retrieve email information, and engage in conversations with an AI chatbot.

The backend of our software primarily utilizes the Flask framework. Flask is user-friendly, highly extensible, and perfectly suited for our project.

2. 安装和配置

packages	version
flask	2.2.3
flask-sqlalchemy	3.0.3

jinja2	3.1.2
openai	0.27.4
pytest	7.3.1
pytest-cov	4.0.0
zmail	0.2.8

```
pip install Flask==2.2.3
pip install Flask-SQLAlchemy==3.0.3
pip install Jinja2==3.1.2
pip install openai==0.27.4
pip install pytest==7.3.1
pip install pytest-cov==4.0.0
pip install zmail
```

3. 快速入门

- It is recommended to use Anaconda to create a virtual environment, then use Python 3.9 as the base, and follow the commands mentioned above to install all the required libraries.
- Download the complete project from GitHub: https://github.com/Barry-Yellow/SoftwareProject-main.git
- Open the project using PyCharm and change the interpreter to the created virtual environment. Ensure that PostgreSQL database is installed and configured accordingly by modifying the configuration in base.py file to match the settings of your current device.
- Run main.py to start the project.

4. 核心功能和实现

4.1 User management

- Function Description: These methods is used to implement user management functions including login, register, modify, verify by email and so on.
- Implementation Method Code Location: In main.py: register_by_email(), login_by_email(), login(), modify().
- Important Notes: In the user management section, we have implemented the use of email authentication to register users, and then users can log in using their email or registered password. After logging in, users can apply to modify their identity

information, but they need to enter a password to successfully modify it.

4.2 post comments & get comments

- Function Description: Users can search for courses and click to enter the comment area, where they can post comments or view existing ones.
- Implementation Method Code Location: In main.py: post_comments() & get_comments()
- Important Notes: When viewing and posting comments, we will check the corresponding user information. If the user information does not match the information in the database, we will refuse access. When viewing comments, we will reply with a JSON string that contains all comments for a certain course.

4.3 select class

- Function Description: This function returns the generated course selection scheme and characteristic values by entering a list of selected courses.
- Implementation Method Code Location: /database/algorithm.py
- Important Notes: The backtracking search algorithm is used here. When the amount of data is too large, the response speed may be slowed down. Appropriate optimization can be done.

4.4 view all courses

- Function Description: Get a list of all the classes.
- Implementation Method Code Location: /database/Course.py
- Important Notes:Please modify your database connection url correctly in the base.py to ensure that this function works properly.

4.5 AI for Email

- Function Description: Integrate the prompt word project of chatGPT Promat to filter and summarize users' email information based on their personal characteristics and preferences.
- Implementation Method Code Location: /module/receivedMail.py
- Important Notes: Please enable the POP/SMTP function before using the function. The

response effect varies according to the user's openai account level.

4.6 AI for conversation

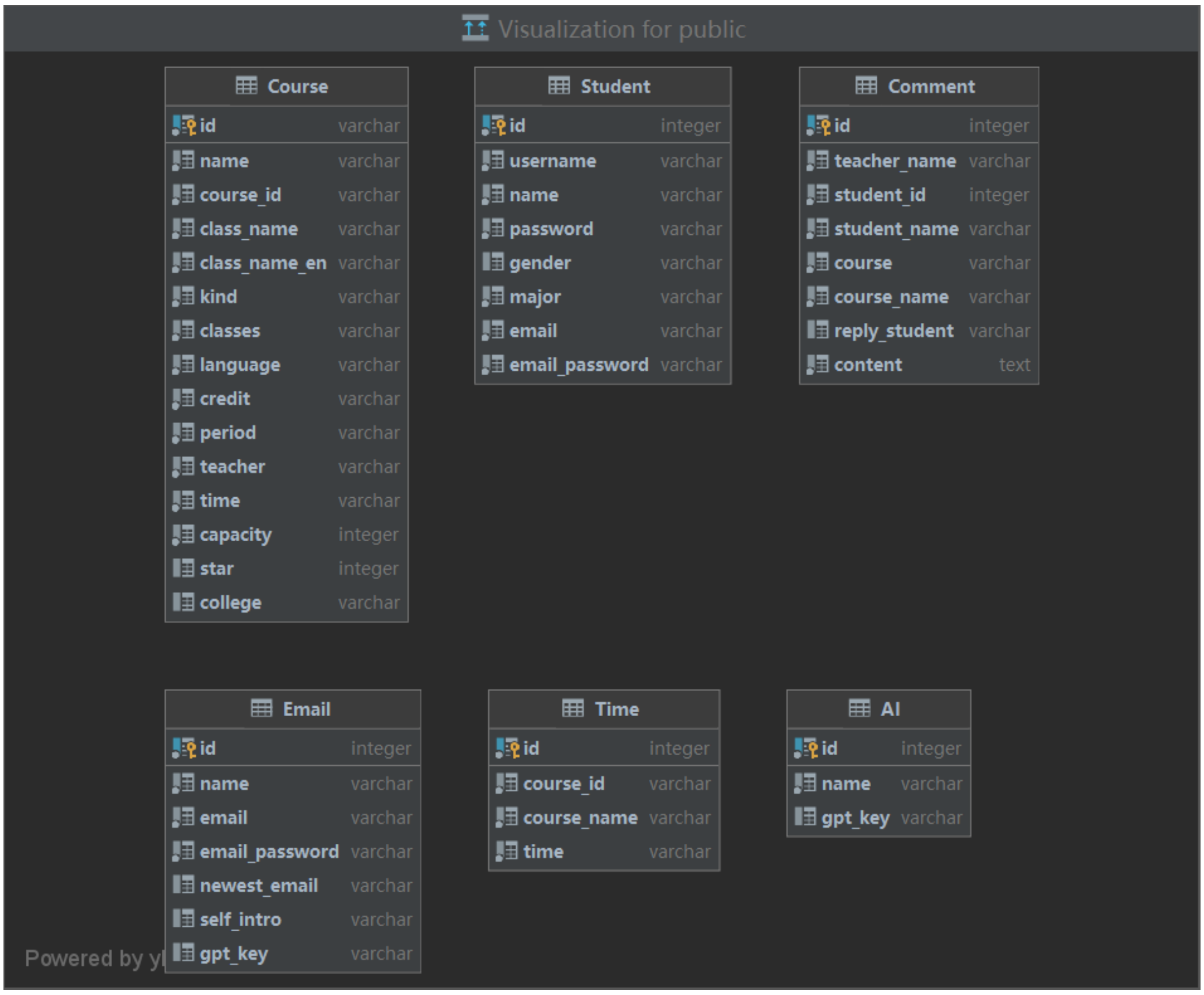
- Function Description: First batch chatGPT app on Android, combined with the openai api, allows users to interact with AI.
- Implementation Method Code Location: /module/ai.py
- Important Notes: The response will vary depending on the user's openai account level.

5. 数据库集成

When creating tables, the **Base = declarative_base()** statement creates a base class provided by **SQLAlchemy**. This class has various functionalities and attributes provided by SQLAlchemy, such as mapping database tables and defining model class methods. By **inheriting** this base class, we can utilize these functionalities and attributes in our model classes to establish a mapping between them and the database tables.

In the Flask framework, interacting with databases requires the use of the **Flask-SQLAlchemy** extension. **SQLAlchemy()** is a class, and by creating an instance of this class named db, we can utilize its methods and attributes to perform database-related operations, such as creating tables, querying data, inserting records, updating records, and more.

Specific methods and interfaces for querying, modifying, and other operations can be found in **base.py**.



Course	
id	varchar
name	varchar
course_id	varchar
class_name	varchar
class_name_en	varchar
kind	varchar
classes	varchar
language	varchar
credit	varchar
period	varchar
teacher	varchar
time	varchar
capacity	integer
star	integer
college	varchar

Student	
id	integer
username	varchar
name	varchar
password	varchar
gender	varchar
major	varchar
email	varchar
email_password	varchar

Comment	
id	integer
teacher_name	varchar
student_id	integer
student_name	varchar
course	varchar
course_name	varchar
reply_student	varchar
content	text

Email	
id	integer
name	varchar
email	varchar
email_password	varchar
newest_email	varchar
self_intro	varchar
gpt_key	varchar

Time	
id	integer
course_id	varchar
course_name	varchar
time	varchar

AI	
id	integer
name	varchar
gpt_key	varchar

6. API 文档

如果日程软件提供 API 接口，提供相应的 API 文档。

列出所有可用的 API 端点、请求和响应格式，并提供使用示例。

7. 测试工具与方法

7.1 tools:

- Unit Test & IT: pytest & coverage
- End-to-End Test: postman & frontend app

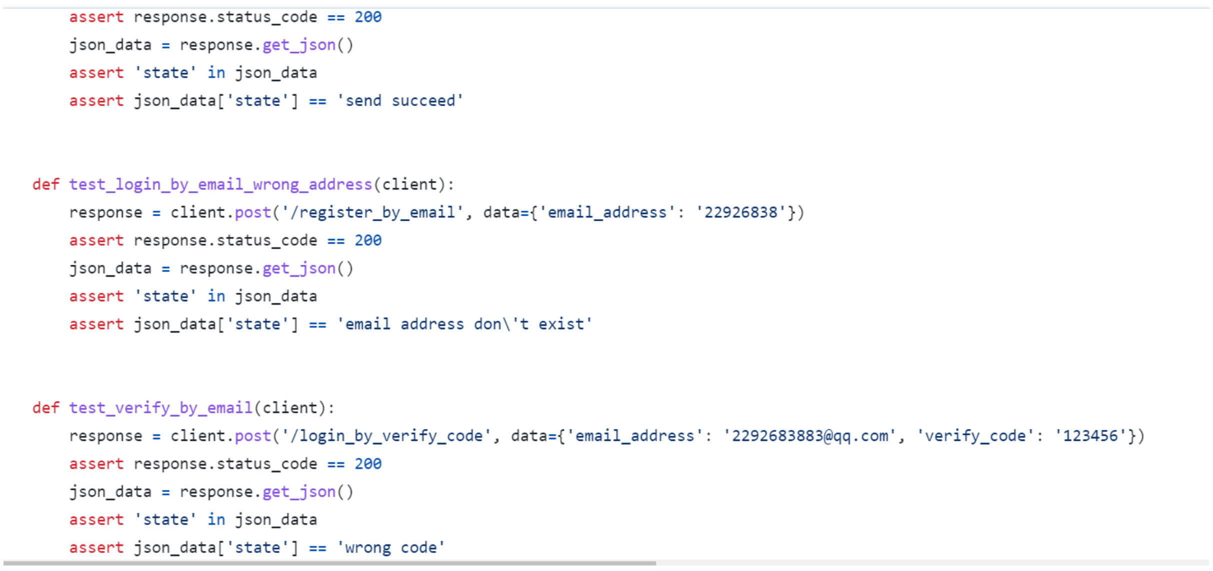
7.2 Unit Test & IT

We use pytest to test each branch and the corresponding url route. In each test we set a list of parameters and then GET/POST these data in json to the route we want to test. After the app return response, we get the data from it and assert if it's in expected.

As for coverage report, we use coverage to generate a report. The coverage report can be seen in the GitHub link https://github.com/Barry-Yellow/test/blob/8b7c8fb476b359e8fa57e5d6f447d6cde94625b8/TeamProject/htmlcov/index.html

source code of test： link :https://github.com/Barry-Yellow/test.git

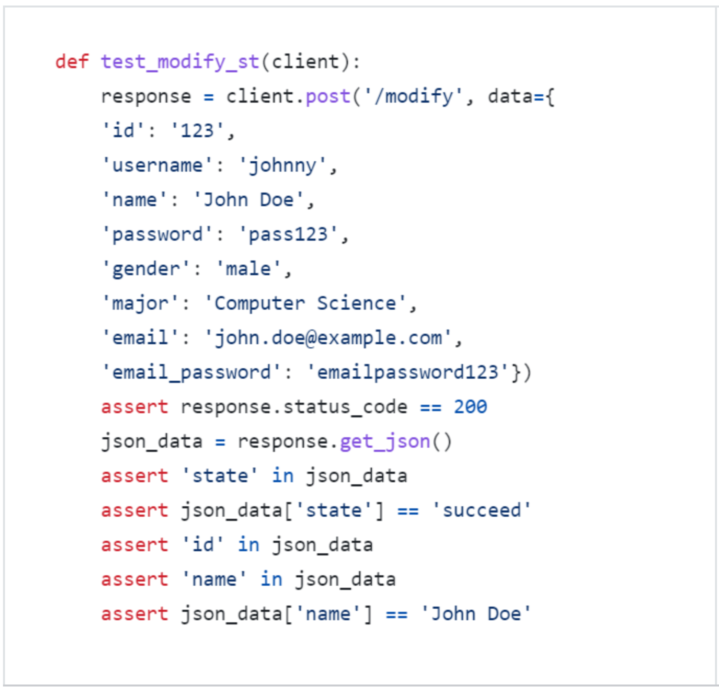
code screenshot:



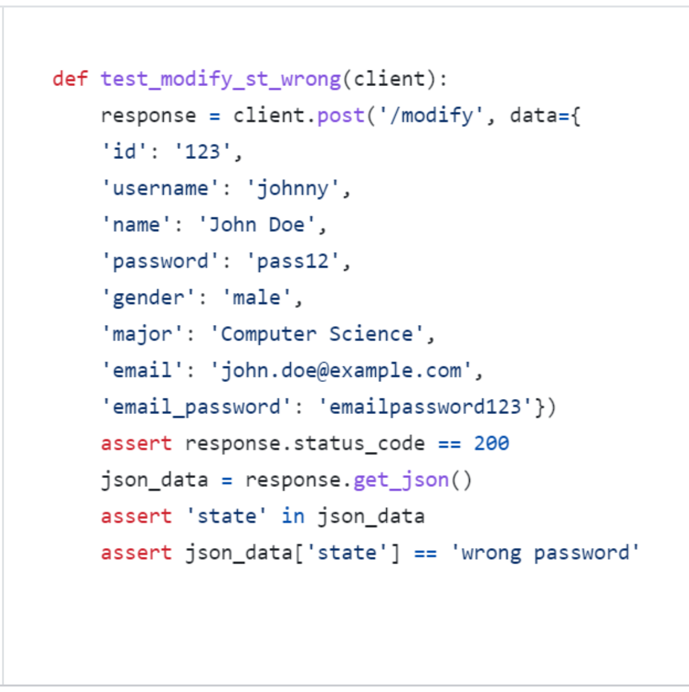
```
        assert response.status_code == 200
        json_data = response.get_json()
        assert 'state' in json_data
        assert json_data['state'] == 'send succeed'


    def test_login_by_email_wrong_address(client):
        response = client.post('/register_by_email', data={'email_address': '22926838'})
        assert response.status_code == 200
        json_data = response.get_json()
        assert 'state' in json_data
        assert json_data['state'] == 'email address don\'t exist'


    def test_verify_by_email(client):
        response = client.post('/login_by_verify_code', data={'email_address': '2292683883@qq.com', 'verify_code': '123456'})
        assert response.status_code == 200
        json_data = response.get_json()
        assert 'state' in json_data
        assert json_data['state'] == 'wrong code'
```



```
def test_modify_st(client):
    response = client.post('/modify', data={
    'id': '123',
    'username': 'johnny',
    'name': 'John Doe',
    'password': 'pass123',
    'gender': 'male',
    'major': 'Computer Science',
    'email': 'john.doe@example.com',
    'email_password': 'emailpassword123'})
    assert response.status_code == 200
    json_data = response.get_json()
    assert 'state' in json_data
    assert json_data['state'] == 'succeed'
    assert 'id' in json_data
    assert 'name' in json_data
    assert json_data['name'] == 'John Doe'
```



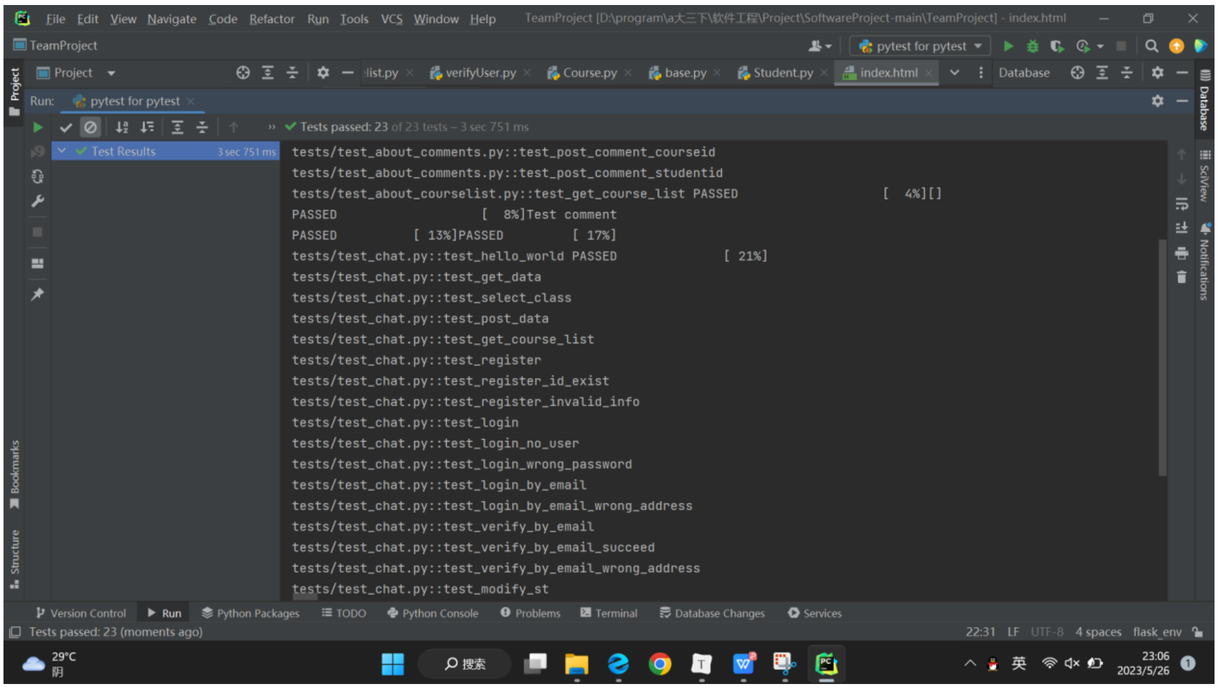
```
def test_modify_st_wrong(client):
    response = client.post('/modify', data={
    'id': '123',
    'username': 'johnny',
    'name': 'John Doe',
    'password': 'pass12',
    'gender': 'male',
    'major': 'Computer Science',
    'email': 'john.doe@example.com',
    'email_password': 'emailpassword123'})
    assert response.status_code == 200
    json_data = response.get_json()
    assert 'state' in json_data
    assert json_data['state'] == 'wrong password'
```

```python
def test_modify_st(client):
    response = client.post('/modify', data={
    'id': '123',
    'username': 'johnny',
    'name': 'John Doe',
    'password': 'pass123',
    'gender': 'male',
    'major': 'Computer Science',
    'email': 'john.doe@example.com',
    'email_password': 'emailpassword123'})
    assert response.status_code == 200
    json_data = response.get_json()
    assert 'state' in json_data
    assert json_data['state'] == 'succeed'
    assert 'id' in json_data
    assert 'name' in json_data
    assert json_data['name'] == 'John Doe'
```

```python
def test_login_no_user(client):
    response = client.post('/login', data={'id': '1234',
    assert response.status_code == 200
    json_data = response.get_json()
    assert 'state' in json_data
    assert json_data['state'] == 'no such user'


def test_login_wrong_password(client):
    response = client.post('/login', data={'id': '123',
    assert response.status_code == 200
    json_data = response.get_json()
    assert 'state' in json_data
    assert json_data['state'] == 'wrong password'
```

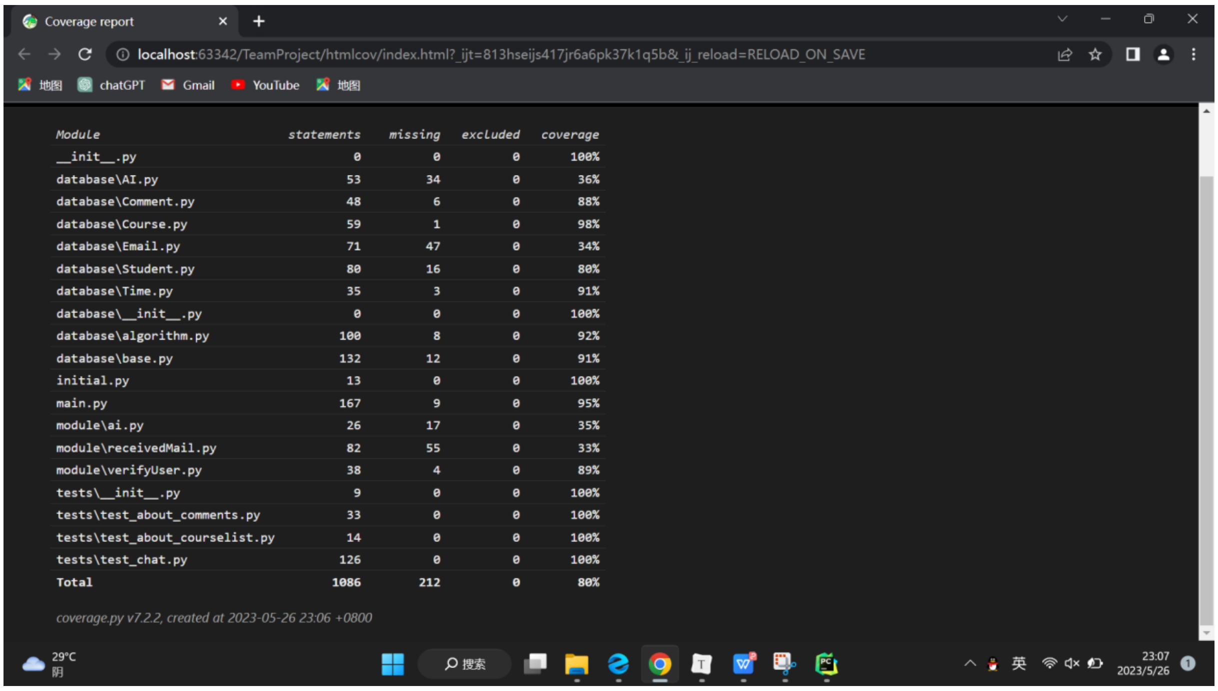
result screenshot:



Tests passed: 23 of 23 tests – 3 sec 751 ms

```
tests/test_about_comments.py::test_post_comment_courseid
tests/test_about_comments.py::test_post_comment_studentid
tests/test_about_courselist.py::test_get_course_list PASSED                [  4%][]
PASSED                  [  8%]Test comment
PASSED            [ 13%]PASSED           [ 17%]
tests/test_chat.py::test_hello_world PASSED                [ 21%]
tests/test_chat.py::test_get_data
tests/test_chat.py::test_select_class
tests/test_chat.py::test_post_data
tests/test_chat.py::test_get_course_list
tests/test_chat.py::test_register
tests/test_chat.py::test_register_id_exist
tests/test_chat.py::test_register_invalid_info
tests/test_chat.py::test_login
tests/test_chat.py::test_login_no_user
tests/test_chat.py::test_login_wrong_password
tests/test_chat.py::test_login_by_email
tests/test_chat.py::test_login_by_email_wrong_address
tests/test_chat.py::test_verify_by_email
tests/test_chat.py::test_verify_by_email_succeed
tests/test_chat.py::test_verify_by_email_wrong_address
tests/test_chat.py::test_modify_st
```

Tests passed: 23 (moments ago)



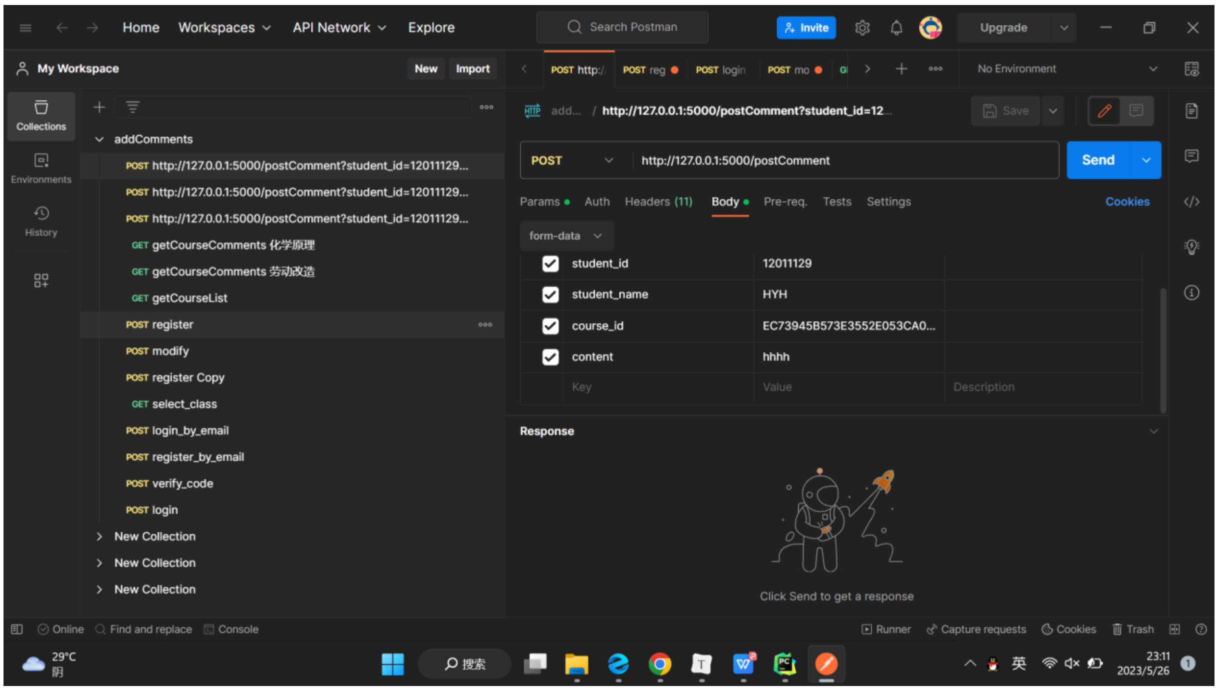
```
Module                        statements  missing  excluded  coverage
__init__.py                            0        0         0      100%
database\AI.py                        53       34         0       36%
database\Comment.py                   48        6         0       88%
database\Course.py                    59        1         0       98%
database\Email.py                     71       47         0       34%
database\Student.py                   80       16         0       80%
database\Time.py                      35        3         0       91%
database\__init__.py                   0        0         0      100%
database\algorithm.py                100        8         0       92%
database\base.py                     132       12         0       91%
initial.py                            13        0         0      100%
main.py                              167        9         0       95%
module\ai.py                          26       17         0       35%
module\receivedMail.py                82       55         0       33%
module\verifyUser.py                  38        4         0       89%
tests\__init__.py                      9        0         0      100%
tests\test_about_comments.py          33        0         0      100%
tests\test_about_courselist.py        14        0         0      100%
tests\test_chat.py                   126        0         0      100%
Total                               1086      212         0       80%

coverage.py v7.2.2, created at 2023-05-26 23:06 +0800
```

The coverage report can be seen in the GitHub link https://github.com/Barry-Yellow/test/blob/8b7c8fb476b359e8fa57e5d6f447d6cde94625b8/TeamProject/htmlcov/index.html

7.3 End-to-End test

We wrote URLs in postman, so that we can simply configure the parameters in HTTP body or somewhere in some certain format. As shown below, we can send GET/POST method to test our interface, and then check the response.



```
My Workspace
addComments
  POST http://127.0.0.1:5000/postComment?student_id=12011129...
  POST http://127.0.0.1:5000/postComment?student_id=12011129...
  POST http://127.0.0.1:5000/postComment?student_id=12011129...
  GET getCourseComments 化学原理
  GET getCourseComments 劳动改造
  GET getCourseList
  POST register
  POST modify
  POST register Copy
  GET select_class
  POST login_by_email
  POST register_by_email
  POST verify_code
  POST login
New Collection
New Collection
New Collection

POST  http://127.0.0.1:5000/postComment
Body: form-data
  student_id     12011129
  student_name   HYH
  course_id      EC73945B573E3552E053CA0...
  content        hhhh

Response
Click Send to get a response
```

8. 构建与部署

8.1 Build

Part I

The packaging tool I use is PyInstaller. PyInstaller is a tool used to package Python applications into standalone executable files. It can package Python code and its dependencies into a single executable file, eliminating the need to install the Python interpreter or other dependencies on the user's computer.

With PyInstaller, you can convert Python applications into executable files that can run on different platforms such as Windows, Mac, and Linux. It supports packaging Python code into a single executable file or folder, and it can automatically handle the required dependencies for running on different systems.

Part II

The command for building our program is:

`pyinstaller main.py -F`PyInstaller performs the following tasks during the build process:

1. Import Analysis: PyInstaller analyzes the Python source code or script to determine which modules are imported and used.

2. Module Collection: Based on the import analysis results, PyInstaller collects all the modules and related resource files that are being used.

3. Module Analysis: The collected modules are further analyzed by PyInstaller to understand the dependencies between the modules.

4. Handling Data Files: PyInstaller processes the data files used in the application, such as images, audio, or other resource files.

5. Generating the Executable: In the final stage of the build process, PyInstaller packages all the modules and resource files together to generate a standalone executable file.

6. Resolving Dependencies: PyInstaller automatically resolves the dependencies required by the application, including Python modules, third-party libraries, and other resource files.

7. Generating the Bootloader Script: PyInstaller also generates a bootloader script that is responsible for starting the application and setting up the necessary runtime environment.

After a successful build using PyInstaller (using the aforementioned command), it will generate two folders in the current directory: "build" and "dist".

The "build" folder contains information related to the build process, such as temporary files, logs, and intermediate artifacts generated during the build.

The "dist" folder, on the other hand, contains the final output of the build process. It typically includes the generated standalone executable file, along with any required resources or dependencies needed for the application to run independently on the target system.

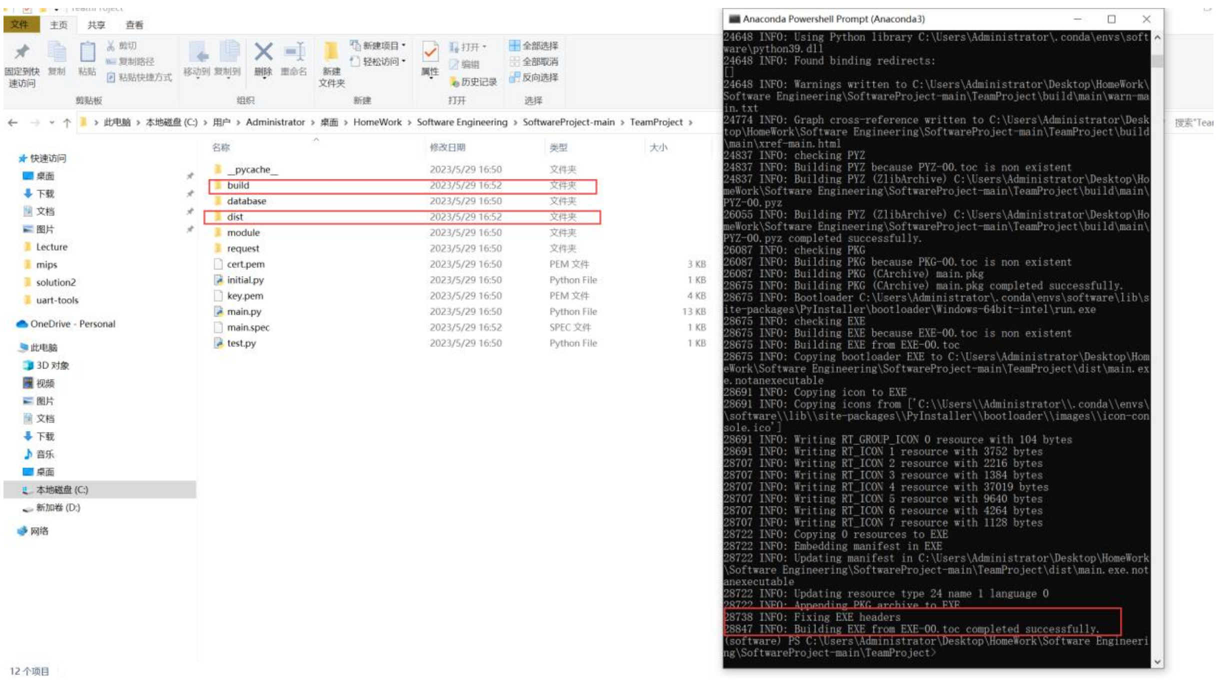
It's worth noting that the specific contents of the "build" and "dist" folders may vary depending on the options and configurations used during the PyInstaller build process.

Part III

The command for building our program is:

```
pyinstaller main.py -F
```

No additional compilation script is required. This is a screen shot of a successful compilation



名称	修改日期	类型	大小
__pycache__	2023/5/29 16:50	文件夹	
build	2023/5/29 16:52	文件夹	
database	2023/5/29 16:50	文件夹	
dist	2023/5/29 16:52	文件夹	
module	2023/5/29 16:50	文件夹	
request	2023/5/29 16:50	文件夹	
cert.pem	2023/5/29 16:50	PEM 文件	3 KB
initial.py	2023/5/29 16:50	Python File	1 KB
key.pem	2023/5/29 16:50	PEM 文件	4 KB
main.py	2023/5/29 16:50	Python File	13 KB
main.spec	2023/5/29 16:52	SPEC 文件	1 KB
test.py	2023/5/29 16:50	Python File	1 KB

```
24648 INFO: Using Python library C:\Users\Administrator\.conda\envs\soft
ware\python39.dll
24648 INFO: Found binding redirects:
[]
24648 INFO: Warnings written to C:\Users\Administrator\Desktop\HomeWork\
Software Engineering\SoftwareProject-main\TeamProject\build\main\warn-ma
in.txt
24774 INFO: Graph cross-reference written to C:\Users\Administrator\Desk
top\HomeWork\Software Engineering\SoftwareProject-main\TeamProject\build
\main\xref-main.html
24837 INFO: checking PYZ
24837 INFO: Building PYZ because PYZ-00.toc is non existent
24837 INFO: Building PYZ (ZlibArchive) C:\Users\Administrator\Desktop\Ho
meWork\Software Engineering\SoftwareProject-main\TeamProject\build\main\
PYZ-00.pyz
26055 INFO: Building PYZ (ZlibArchive) C:\Users\Administrator\Desktop\Ho
meWork\Software Engineering\SoftwareProject-main\TeamProject\build\main\
PYZ-00.pyz completed successfully.
26087 INFO: checking PKG
26087 INFO: Building PKG because PKG-00.toc is non existent
26087 INFO: Building PKG (CArchive) main.pkg
28675 INFO: Building PKG (CArchive) main.pkg completed successfully.
28675 INFO: Bootloader C:\Users\Administrator\.conda\envs\software\lib\s
ite-packages\PyInstaller\bootloader\Windows-64bit-intel\run.exe
28675 INFO: checking EXE
28675 INFO: Building EXE because EXE-00.toc is non existent
28675 INFO: Building EXE from EXE-00.toc
28675 INFO: Copying bootloader EXE to C:\Users\Administrator\Desktop\Hom
eWork\Software Engineering\SoftwareProject-main\TeamProject\dist\main.ex
e.notanexecutable
28691 INFO: Copying icon to EXE
28691 INFO: Copying icons from ['C:\\Users\\Administrator\\.conda\\envs\
\software\\lib\\site-packages\\PyInstaller\\bootloader\\images\\icon-con
sole.ico']
28691 INFO: Writing RT_GROUP_ICON 0 resource with 104 bytes
28691 INFO: Writing RT_ICON 1 resource with 3752 bytes
28707 INFO: Writing RT_ICON 2 resource with 2216 bytes
28707 INFO: Writing RT_ICON 3 resource with 1384 bytes
28707 INFO: Writing RT_ICON 4 resource with 37019 bytes
28707 INFO: Writing RT_ICON 5 resource with 9640 bytes
28707 INFO: Writing RT_ICON 6 resource with 4264 bytes
28707 INFO: Writing RT_ICON 7 resource with 1128 bytes
28722 INFO: Copying 0 resources to EXE
28722 INFO: Embedding manifest in EXE
28722 INFO: Updating manifest in C:\Users\Administrator\Desktop\HomeWork
\Software Engineering\SoftwareProject-main\TeamProject\dist\main.exe.not
anexecutable
28722 INFO: Updating resource type 24 name 1 language 0
28722 INFO: Appending PKG archive to EXE
28738 INFO: Fixing EXE headers
28847 INFO: Building EXE from EXE-00.toc completed successfully.
(software) PS C:\Users\Administrator\Desktop\HomeWork\Software Engineeri
ng\SoftwareProject-main\TeamProject>
```

8.2 Deployment

Part I

The project containerization technology I'm using is Docker. Docker is a containerization deployment technology that provides a lightweight solution for packaging applications and their dependencies into independent containers. This allows them to be deployed and run in different environments without concerns about environmental differences.

With Docker, you can create an image that includes the application, its related configurations, libraries, and dependencies. This image can be deployed and run in any Docker-supported environment, including development machines, testing environments, and production servers.

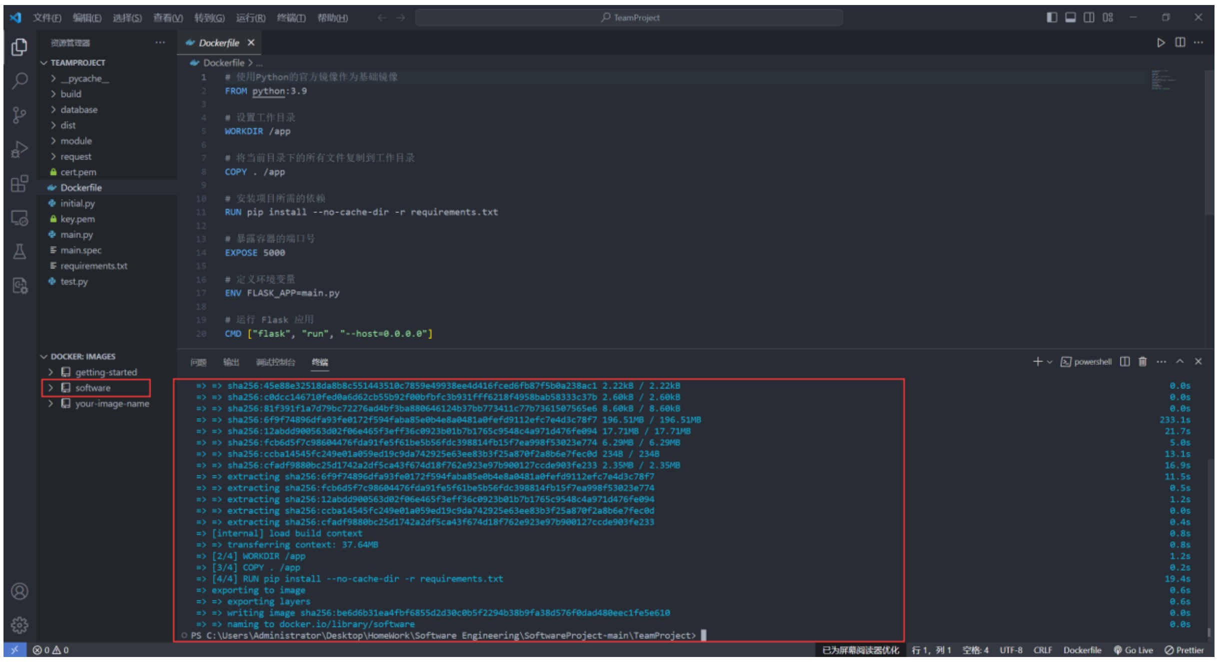
Docker containers provide an isolated runtime environment that ensures consistent behavior of the application across different environments.

Part II

This is the Dockerfile I use, before using docker, I need to generates a list of dependencies for a python project. The command I used to generate the requrements.txt is：

```
pipreqs . --encoding=utf8
```

Part III



```
# 使用Python的官方镜像作为基础镜像
FROM python:3.9

# 设置工作目录
WORKDIR /app

# 将当前目录下的所有文件复制到工作目录
COPY . /app

# 安装项目所需的依赖
RUN pip install --no-cache-dir -r requirements.txt

# 暴露容器的端口号
EXPOSE 5000

# 定义环境变量
ENV FLASK_APP=main.py

# 运行 Flask 应用
CMD ["flask", "run", "--host=0.0.0.0"]
```

```
 => => sha256:45e88e32518da8b8c551443510c7859e49938ee4d416fced6fb87f5b0a238ac1 2.22kB / 2.22kB        0.0s
 => => sha256:c0dcc146710fed0a6d62cb55b92f00bfbfc3b931fff6218f4958bab58333c37b 2.60kB / 2.60kB        0.0s
 => => sha256:81f391f1a7d79bc72276ad4bf3ba880646124b37bb773411c77b7361507565e6 8.60kB / 8.60kB        0.0s
 => => sha256:6f9f74896dfa93fe0172f594faba85e0b4e8a0481a0fefd9112efc7e4d3c78f7 196.51MB / 196.51MB  233.1s
 => => sha256:12abdd900563d02f06e465f3eff36c0923b01b7b1765c9548c4a971d476fe094 17.71MB / 17.71MB     21.7s
 => => sha256:fcb6d5f7c98604476fda91fe5f61be5b56fdc398814fb15f7ea998f53023e774 6.29MB / 6.29MB        5.0s
 => => sha256:ccba14545fc249e01a059ed19c9da742925e63ee83b3f25a870f2a8b6e7fec0d 234B / 234B           13.1s
 => => sha256:cfadf9880bc25d1742a2df5ca43f674d18f762e923e97b900127ccde903fe233 2.35MB / 2.35MB       16.9s
 => => extracting sha256:6f9f74896dfa93fe0172f594faba85e0b4e8a0481a0fefd9112efc7e4d3c78f7            11.5s
 => => extracting sha256:fcb6d5f7c98604476fda91fe5f61be5b56fdc398814fb15f7ea998f53023e774             0.5s
 => => extracting sha256:12abdd900563d02f06e465f3eff36c0923b01b7b1765c9548c4a971d476fe094             1.2s
 => => extracting sha256:ccba14545fc249e01a059ed19c9da742925e63ee83b3f25a870f2a8b6e7fec0d             0.0s
 => => extracting sha256:cfadf9880bc25d1742a2df5ca43f674d18f762e923e97b900127ccde903fe233             0.4s
 => [internal] load build context                                                                     0.8s
 => => transferring context: 37.64MB                                                                  0.8s
 => [2/4] WORKDIR /app                                                                                1.2s
 => [3/4] COPY . /app                                                                                 0.2s
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt                                         19.4s
 => exporting to image                                                                                0.6s
 => => exporting layers                                                                               0.6s
 => => writing image sha256:be6d6b31ea4fbf6855d2d30c0b5f2294b38b9fa38d576f0dad480eec1fe5e610          0.0s
 => => naming to docker.io/library/software                                                           0.0s
PS C:\Users\Administrator\Desktop\HomeWork\Software Engineering\SoftwareProject-main\TeamProject>
```