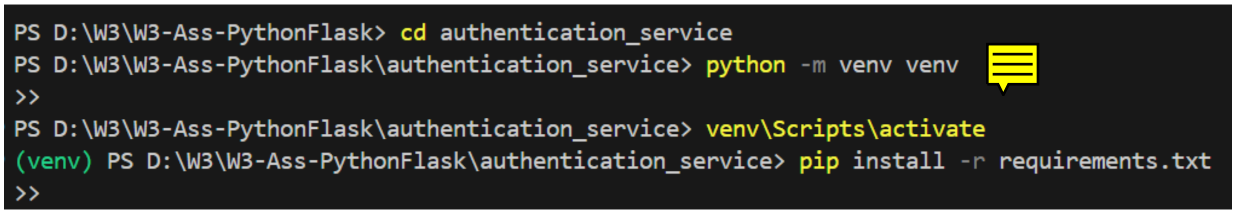
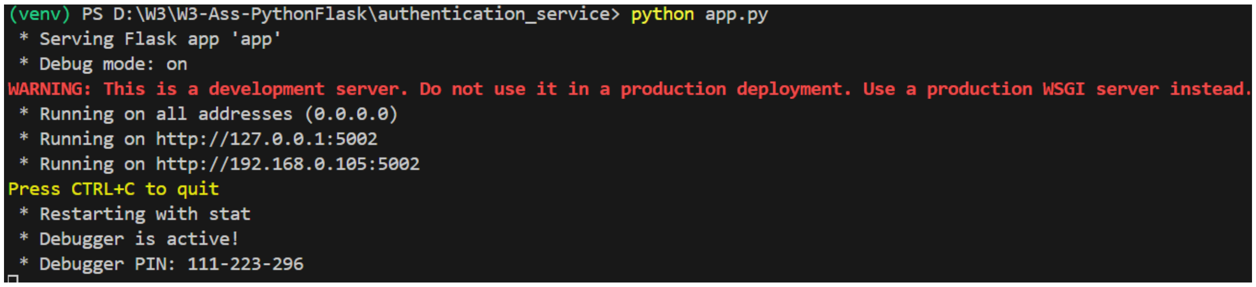
1. **RootFolder/authentication_service**



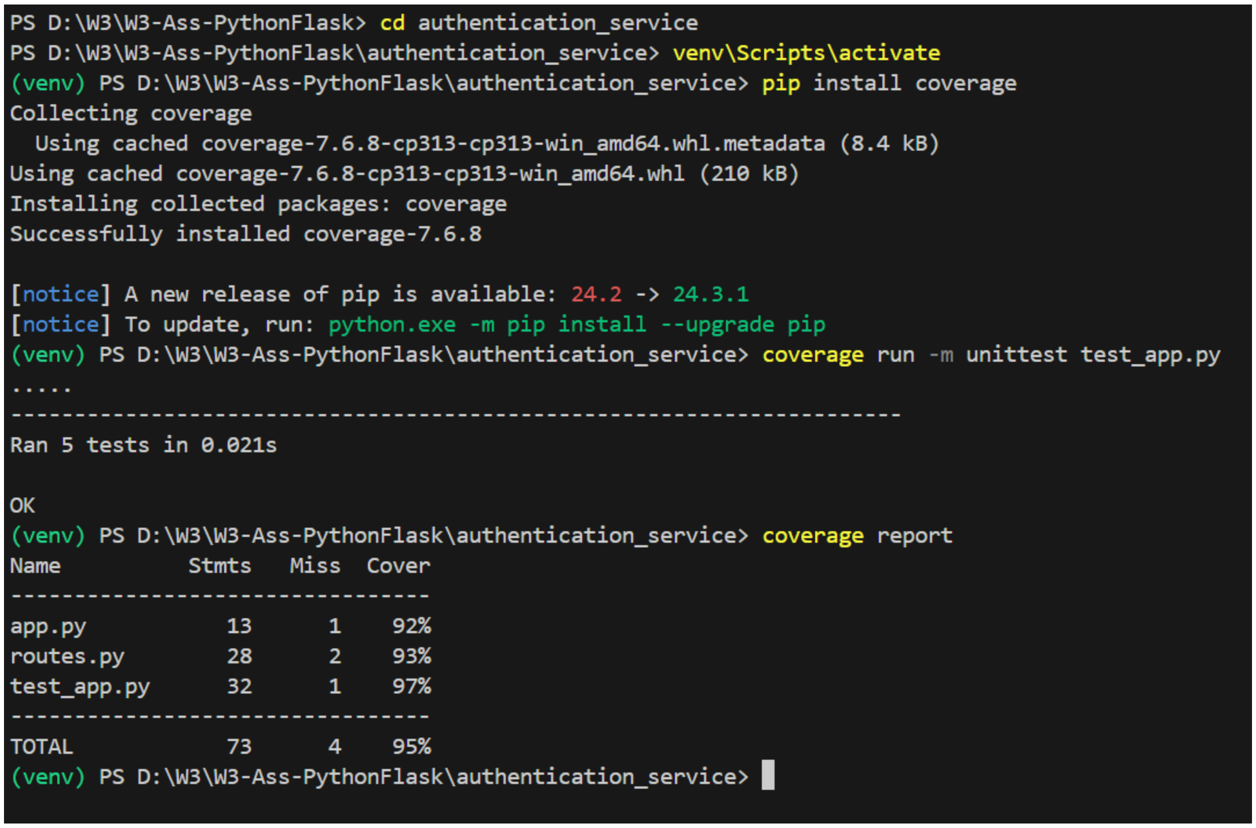
```
PS D:\W3\W3-Ass-PythonFlask> cd authentication_service
PS D:\W3\W3-Ass-PythonFlask\authentication_service> python -m venv venv
>>
PS D:\W3\W3-Ass-PythonFlask\authentication_service> venv\Scripts\activate
(venv) PS D:\W3\W3-Ass-PythonFlask\authentication_service> pip install -r requirements.txt
>>
```

All the requirements will be installed automatically. If you are using Linux instead of **venv\Scripts\activate** type **source venv/bin/activate**



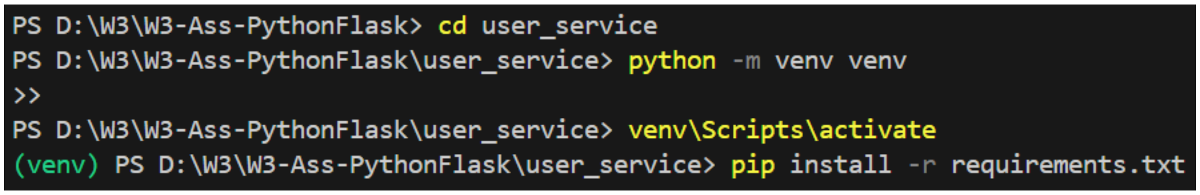
```
(venv) PS D:\W3\W3-Ass-PythonFlask\authentication_service> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://192.168.0.105:5002
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 111-223-296
```

To run the tests for this service, open another terminal while keeping the previous one running, and then follow the instructions shown in this picture.



```
PS D:\W3\W3-Ass-PythonFlask> cd authentication_service
PS D:\W3\W3-Ass-PythonFlask\authentication_service> venv\Scripts\activate
(venv) PS D:\W3\W3-Ass-PythonFlask\authentication_service> pip install coverage
Collecting coverage
  Using cached coverage-7.6.8-cp313-cp313-win_amd64.whl.metadata (8.4 kB)
Using cached coverage-7.6.8-cp313-cp313-win_amd64.whl (210 kB)
Installing collected packages: coverage
Successfully installed coverage-7.6.8

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS D:\W3\W3-Ass-PythonFlask\authentication_service> coverage run -m unittest test_app.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.021s

OK
(venv) PS D:\W3\W3-Ass-PythonFlask\authentication_service> coverage report
Name          Stmts   Miss  Cover
---------------------------------
app.py           13      1    92%
routes.py        28      2    93%
test_app.py      32      1    97%
---------------------------------
TOTAL            73      4    95%
(venv) PS D:\W3\W3-Ass-PythonFlask\authentication_service>
```

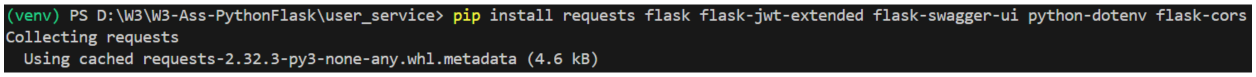
2. RootFolder/user_service



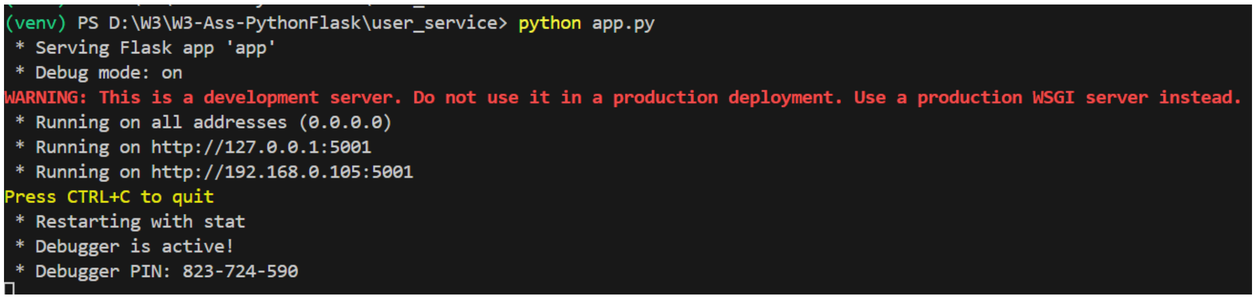
```
PS D:\W3\W3-Ass-PythonFlask> cd user_service
PS D:\W3\W3-Ass-PythonFlask\user_service> python -m venv venv
>>
PS D:\W3\W3-Ass-PythonFlask\user_service> venv\Scripts\activate
(venv) PS D:\W3\W3-Ass-PythonFlask\user_service> pip install -r requirements.txt
```

Then run this command

pip install requests flask flask-jwt-extended flask-swagger-ui python-dotenv flask-cors

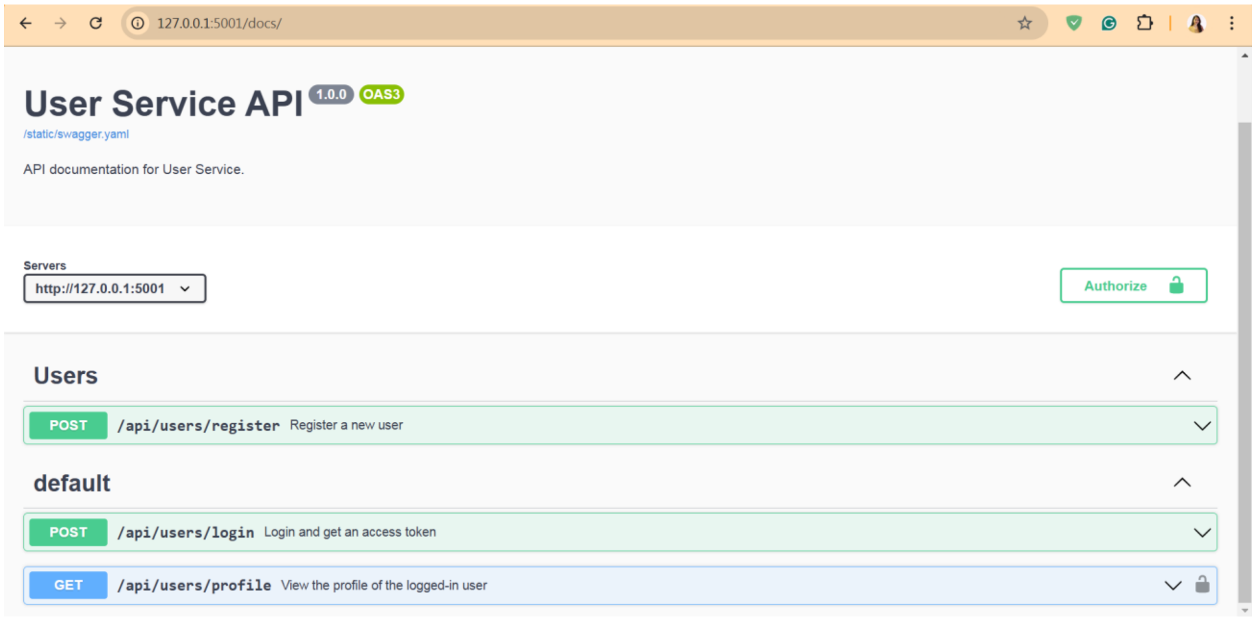


```
(venv) PS D:\W3\W3-Ass-PythonFlask\user_service> pip install requests flask flask-jwt-extended flask-swagger-ui python-dotenv flask-cors
Collecting requests
  Using cached requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
```



```
(venv) PS D:\W3\W3-Ass-PythonFlask\user_service> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.0.105:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 823-724-590
```

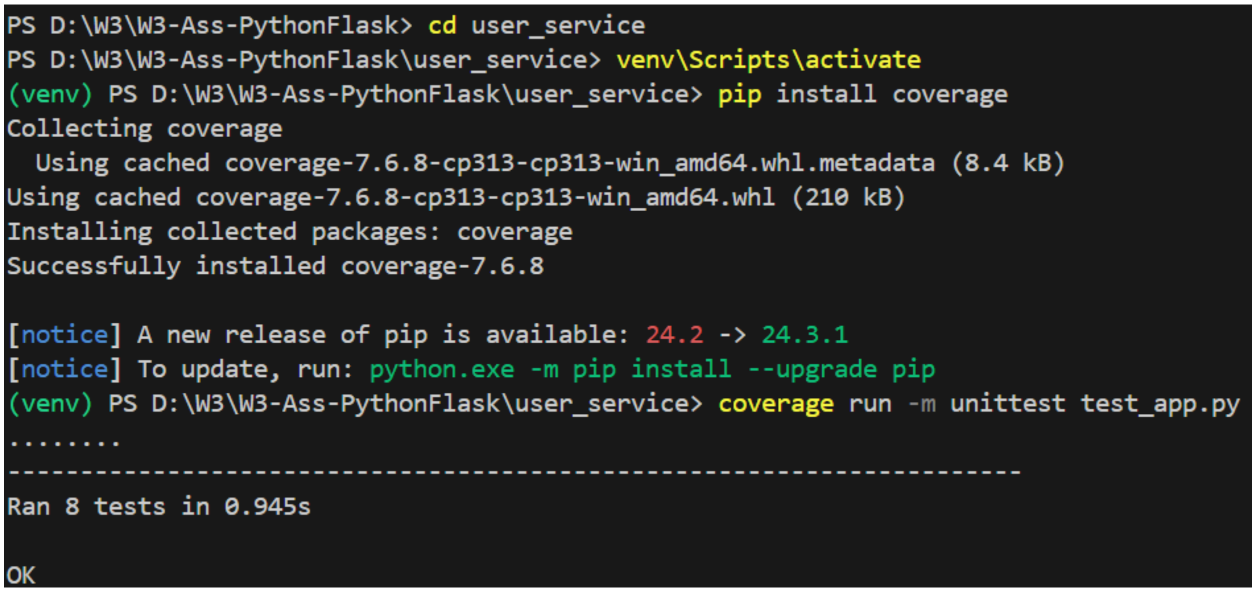
http://127.0.0.1:5001 is your localhost port but for seeing the actual running you will have to go http://127.0.0.1:5001/docs . It will open a Swagger UI which looks like this:


127.0.0.1:5001/docs/
User Service API 1.0.0 OAS3
/static/swagger.yaml
API documentation for User Service.
Servers
http://127.0.0.1:5001
Authorize
Users
POST /api/users/register Register a new user
default
POST /api/users/login Login and get an access token
GET /api/users/profile View the profile of the logged-in user

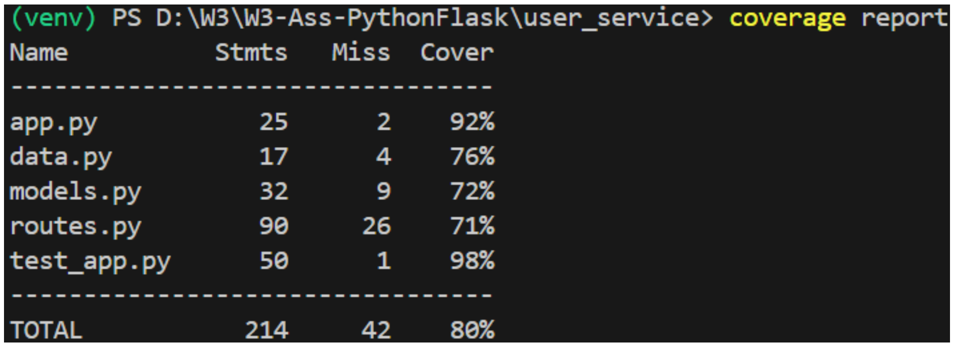
Steps to Check the Project:

1. **Expand Requests**:
   - Click the down arrow of each request to view the corresponding JSON request for POST.
2. **Login and Generate Token**:
   - Provide the email and password to log in.
   - Upon successful login, a token will be generated and displayed in the response body.
3. **Copy the Token**:
   - Copy the token from the response body.
4. **Authorize with the Token**:
   - Look for the **Authorize** button with a lock icon and click on it.
   - Paste the token into the input field provided and authorize.
5. **Test the Requests**:
   - After authorization, your requests will be validated by the **authorize_service**, and you will be able to execute **GET api/users/profile**.

To run the tests for this service, open another terminal while keeping the previous one running, and then follow the instructions shown in this picture.

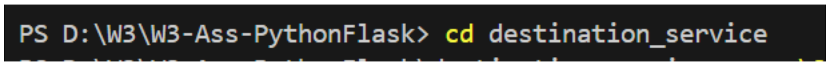


```
PS D:\W3\W3-Ass-PythonFlask> cd user_service
PS D:\W3\W3-Ass-PythonFlask\user_service> venv\Scripts\activate
(venv) PS D:\W3\W3-Ass-PythonFlask\user_service> pip install coverage
Collecting coverage
  Using cached coverage-7.6.8-cp313-cp313-win_amd64.whl.metadata (8.4 kB)
Using cached coverage-7.6.8-cp313-cp313-win_amd64.whl (210 kB)
Installing collected packages: coverage
Successfully installed coverage-7.6.8

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS D:\W3\W3-Ass-PythonFlask\user_service> coverage run -m unittest test_app.py
........
----------------------------------------------------------------------
Ran 8 tests in 0.945s

OK
```

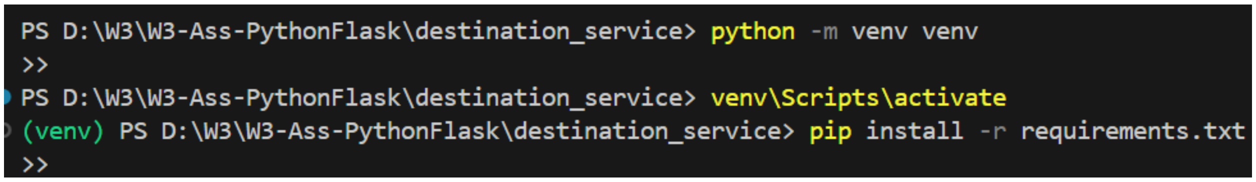


```
(venv) PS D:\W3\W3-Ass-PythonFlask\user_service> coverage report
Name            Stmts   Miss  Cover
-----------------------------------
app.py             25      2    92%
data.py            17      4    76%
models.py          32      9    72%
routes.py          90     26    71%
test_app.py        50      1    98%
-----------------------------------
TOTAL             214     42    80%
```

3. RootFolder/destination_service



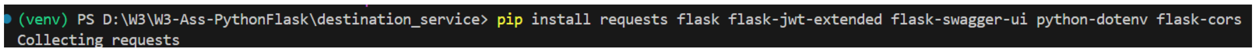
```
PS D:\W3\W3-Ass-PythonFlask> cd destination_service
```



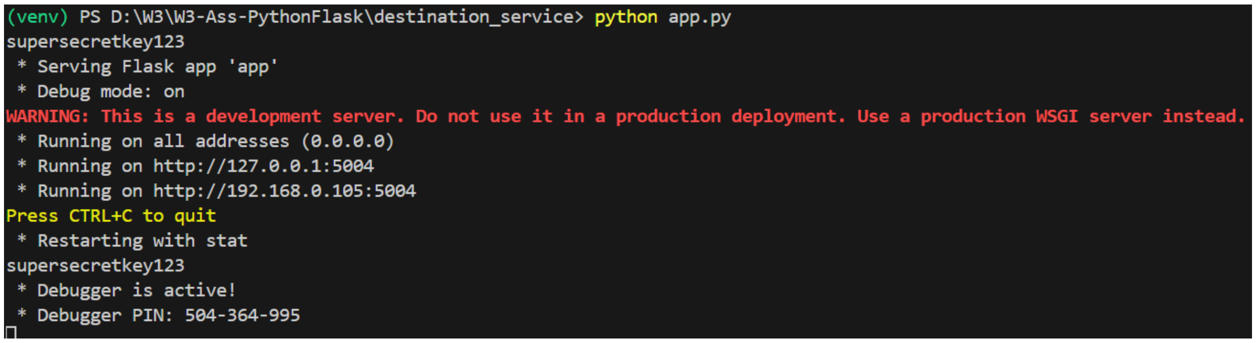
```
PS D:\W3\W3-Ass-PythonFlask\destination_service> python -m venv venv
>>
PS D:\W3\W3-Ass-PythonFlask\destination_service> venv\Scripts\activate
(venv) PS D:\W3\W3-Ass-PythonFlask\destination_service> pip install -r requirements.txt
>>
```

Then run this command

pip install requests flask flask-jwt-extended flask-swagger-ui python-dotenv flask-cors

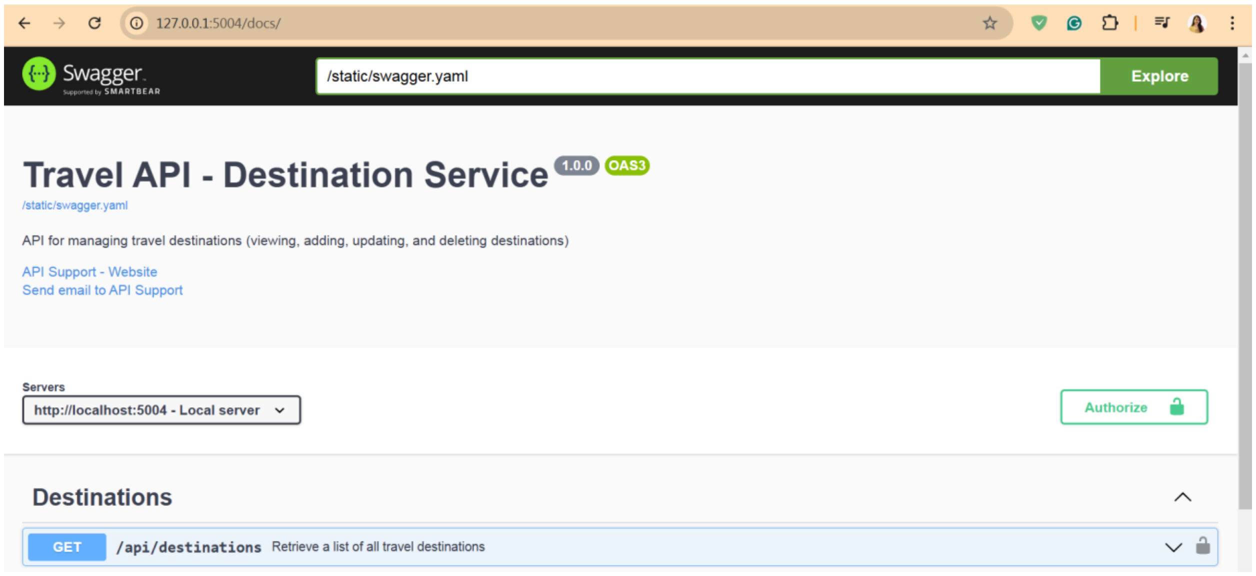


```
(venv) PS D:\W3\W3-Ass-PythonFlask\destination_service> pip install requests flask flask-jwt-extended flask-swagger-ui python-dotenv flask-cors
Collecting requests
```



```
(venv) PS D:\W3\W3-Ass-PythonFlask\destination_service> python app.py
supersecretkey123
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5004
 * Running on http://192.168.0.105:5004
Press CTRL+C to quit
 * Restarting with stat
supersecretkey123
 * Debugger is active!
 * Debugger PIN: 504-364-995
```

http://127.0.0.1:5004 is your localhost port but for seeing the actual running you will have to go http://127.0.0.1:5004/docs . It will open a Swagger UI which looks like this:



127.0.0.1:5004/docs/

Swagger

/static/swagger.yaml

Explore

Travel API - Destination Service 1.0.0 OAS3

/static/swagger.yaml

API for managing travel destinations (viewing, adding, updating, and deleting destinations)

API Support - Website

Send email to API Support

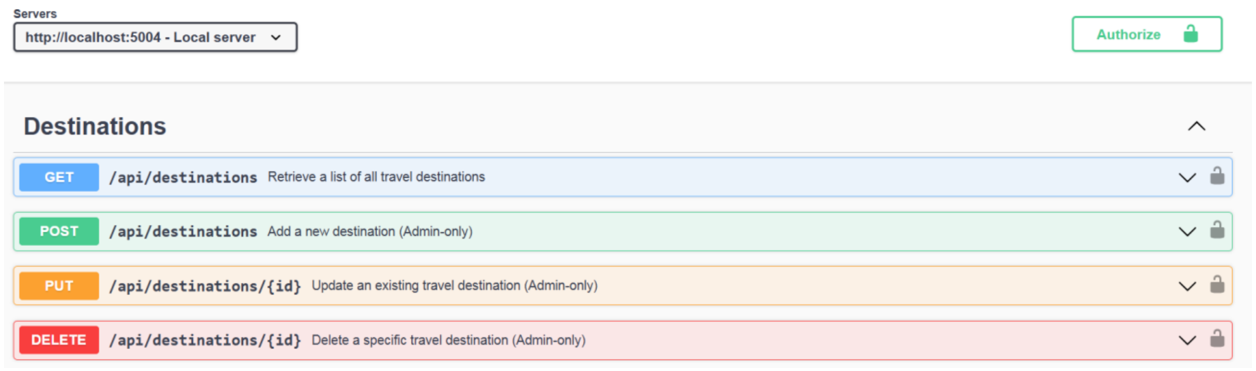
Servers

http://localhost:5004 - Local server

Authorize

Destinations

GET `/api/destinations` Retrieve a list of all travel destinations



Servers

http://localhost:5004 - Local server

Authorize

Destinations

GET `/api/destinations` Retrieve a list of all travel destinations

POST `/api/destinations` Add a new destination (Admin-only)

PUT `/api/destinations/{id}` Update an existing travel destination (Admin-only)

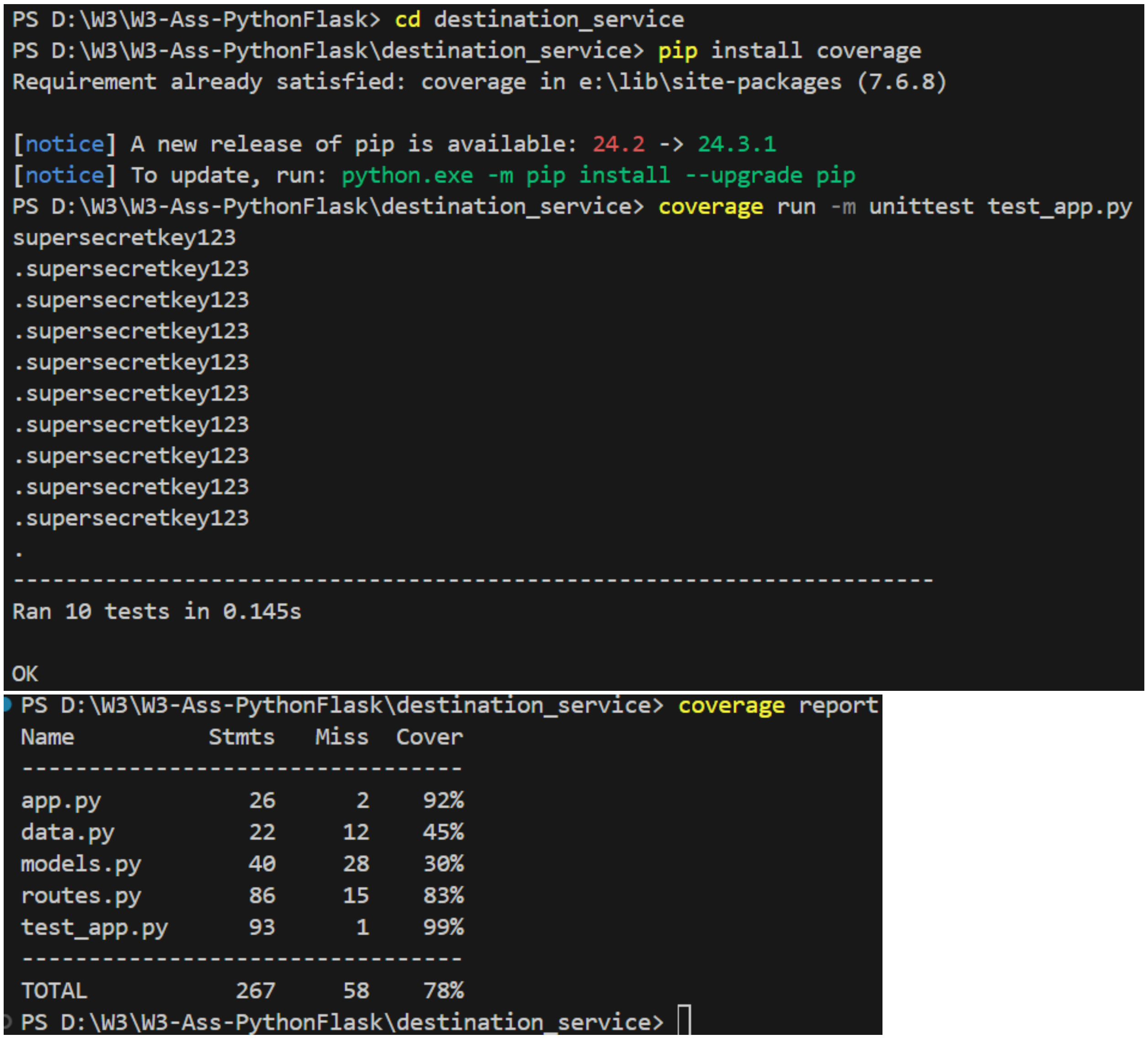
DELETE `/api/destinations/{id}` Delete a specific travel destination (Admin-only)

The first **GET** endpoint for destinations is accessible to all users and does not require authentication. However, the **POST**, **PUT**, and **DELETE** endpoints for managing destinations require admin authorization.

To ensure proper functionality:

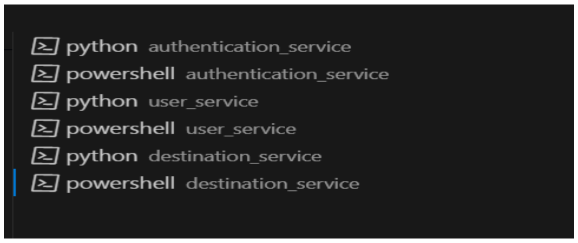
1. Start the **authorization_service** on http://127.0.0.1:5002.
2. Start the **user_service** on http://127.0.0.1:5002/docs.
3. Log in through the **user_service** to generate a valid token.
4. Copy the generated token and paste it into the **Authorize** button (above).
5. Once the token is verified and the user is confirmed as an admin, access to destination management operations will be granted.

To run the tests for this service, open another terminal while keeping the previous one running, and then follow the instructions shown in this picture.



```
PS D:\W3\W3-Ass-PythonFlask> cd destination_service
PS D:\W3\W3-Ass-PythonFlask\destination_service> pip install coverage
Requirement already satisfied: coverage in e:\lib\site-packages (7.6.8)

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS D:\W3\W3-Ass-PythonFlask\destination_service> coverage run -m unittest test_app.py
supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.supersecretkey123
.
----------------------------------------------------------------------
Ran 10 tests in 0.145s

OK
PS D:\W3\W3-Ass-PythonFlask\destination_service> coverage report
Name            Stmts   Miss  Cover
-----------------------------------
app.py             26      2    92%
data.py            22     12    45%
models.py          40     28    30%
routes.py          86     15    83%
test_app.py        93      1    99%
-----------------------------------
TOTAL             267     58    78%
PS D:\W3\W3-Ass-PythonFlask\destination_service>
```

For clarification, when running the project, you need to open three separate terminals and keep them running concurrently to ensure all services are operational. If you are also testing the code for each service, you will need to open a total of six terminals—three for running the services and three additional ones for running the test cases of each service like below.



```
python authentication_service
powershell authentication_service
python user_service
powershell user_service
python destination_service
powershell destination_service
```