Contents

[Discussion (618 words) 2](#_Toc106052832)

[Modules, Libraries and Prerequisites 5](#_Toc106052833)

[Database 6](#_Toc106052834)

[File Tree 7](#_Toc106052835)

[End User Instructions 8](#_Toc106052836)

[Testing 17](#_Toc106052837)

[References 18](#_Toc106052838)

[Appendices 19](#_Toc106052839)

# Discussion (618 words)

NASA requires a highly secure and scalable prototype, considering all three security tenets of the CIA triad. Input sanitisation has been considered, and the system is consistent with PEP8 formatting.

The prototype differs from the original design in the following ways:

It was decided that the smtplib library was not required due to using a one-time-password (OTP) as it was deemed more secure (Thales, N.D.). OTPs provide an additional layer of security by introducing multifactor authentication (MFA), requiring something you have as well as something you know. This exponentially decreases the chance of a user account being compromised as they will need to have access to the device as well as the secret. The QR code used to register a new user must be sent by the admin, separately to the user (Grinberg, 2015).

Another benefit of using MFA is that the lockout / unlock functions originally designed were also deemed unnecessary. However, it could still have been implemented in a more complete version of the software to provide additional security. Brute force and (D)DoS attacks could also be prevented by using network-level security methods, such as a web application firewall (WAF).

Initially, the application was designed so that users would initially create their own user account, which would then need to be verified by an administrator. This allows anyone with access to the application to create their own user account, and potentially a mistake by the admin's onboarding processes could mean the rogue user would have access to substantial amounts of sensitive data. To follow industry recognised practices, it was deemed that the admins should create the user accounts.

Admins also had the right to delete users removed, following best practice guidelines, and enabling users to be reactivated in the future should they need to be, while still prohibiting their access to the system. An extension to the prototype could be to additionally offer a delete function, tied to a company policy to purge them after one year of inactivity, for example.

User notification of changes was also deemed unnecessary due to users having continuous access to their own data and being internal employees, rather than customers or clients. If there would be a reason for NASA to incorporate an interface which would be accessible to the external public, and would therefore their data, then the prototype could include an extended functionality to send email updates to these users whenever changes were made in order to satisfy GDPR.

The attributes of the user and document classes changed considerably. The original "phone" field in the user table was dropped to reduce the attack surface and was already deemed redundant data serving no use to the project. The need for two name fields ("firstName" and "lastName") was also deemed unnecessary for a prototype, and so was consolidated into one. However, this could be extended if more granularity was desired for future iterations. The user table also included an additional "otp\_secret" field to accommodate the OTP function.

The document class also had some changes. The "data" field was introduced to rectify the original oversight that we needed somewhere to store the data binary. The "role" field (originally classification) was changed to an “enum” data type in order to be matched to the role of a user. This was a better idea than the original 'clearance' style classification field which was due to be used and enabled an easier way to share files. Finally, the "key" field is generated once the file is uploaded, combined with the server master key to encrypt, or decrypt the stored files by "MultiFernet". The master key can change after a period of time.

Testing was consistent with what was originally designed and was carried out extensively.

# Modules, Libraries and Prerequisites

The application is built using Flask (The Pallets Projects, N.D.), and the authors assume the user has an internet connection and Python3 installed on their local system.

The following modules and libraries are imported into the program by running **pip3 install –r requirements.txt** from the command line:

Web App

* flask
* flask\_sqlalchemy
* flask\_login
* werkzeug
* cryptography
* pyaml-env
* pymysql
* pytz
* PyQRCode~=1.2.1
* onetimepass~=1.0.1

Testing

* flake8
* pylint
* pytest

# Database

An SQL database has been provided by phpMyAdmin (N.D.) and can be found at: <https://www.phpmyadmin.co>. The create table query can be found in Appendix A.

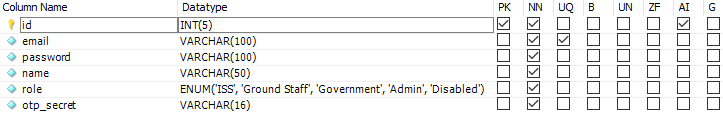


Figure 1: user table

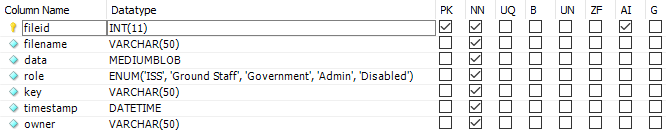


Figure 2: document table

# File Tree

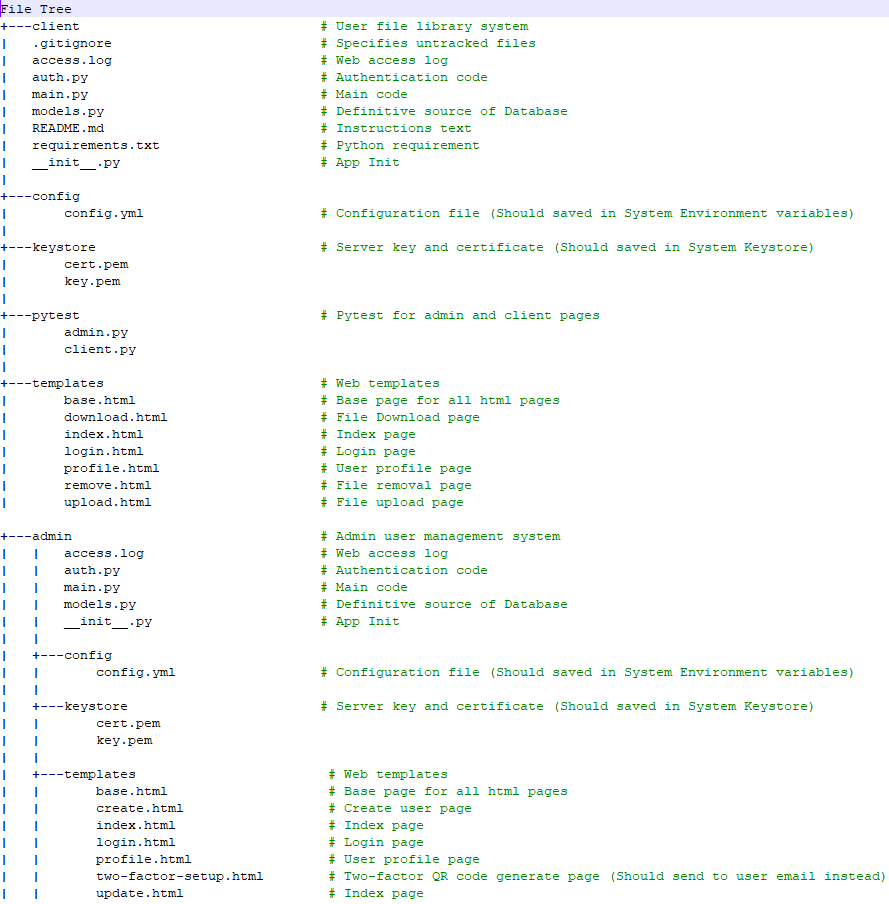


Figure 3: File tree and comments

# End User Instructions

The system has two interfaces. One for admins to create and update user accounts, and one for users to upload, download, and remove documents.

**Admin:**

Setup

Download the application from <https://github.com/ychan2020/SSD2022.git> and run it in the downloaded directory **admin** folder.

1. In the Command Line Interface (CLI), cd [downloaded directory path]\admin

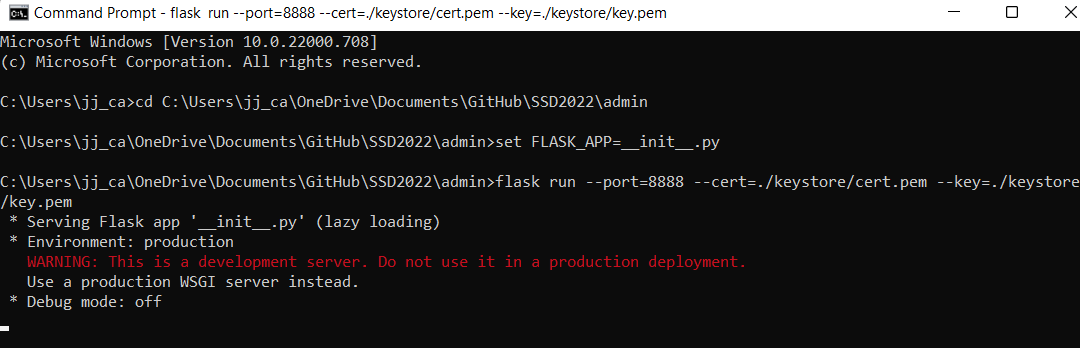


Figure 4: Running application

1. Set FLASK\_APP=\_\_init\_\_.py (Windows) or Set FLASK\_APP=\_\_init\_\_.py (Linux)
2. flask run --port=8888 –-cert=./keystore/cert.pem --key=./keystore/key.pem
3. Open the Admin page <https://localhost:8888/>

Login

1. On the webpage, enter the Admin email username: [admin@test.com](mailto:admin@test.com)
2. The default password: P@ssw0rd
3. Press Login

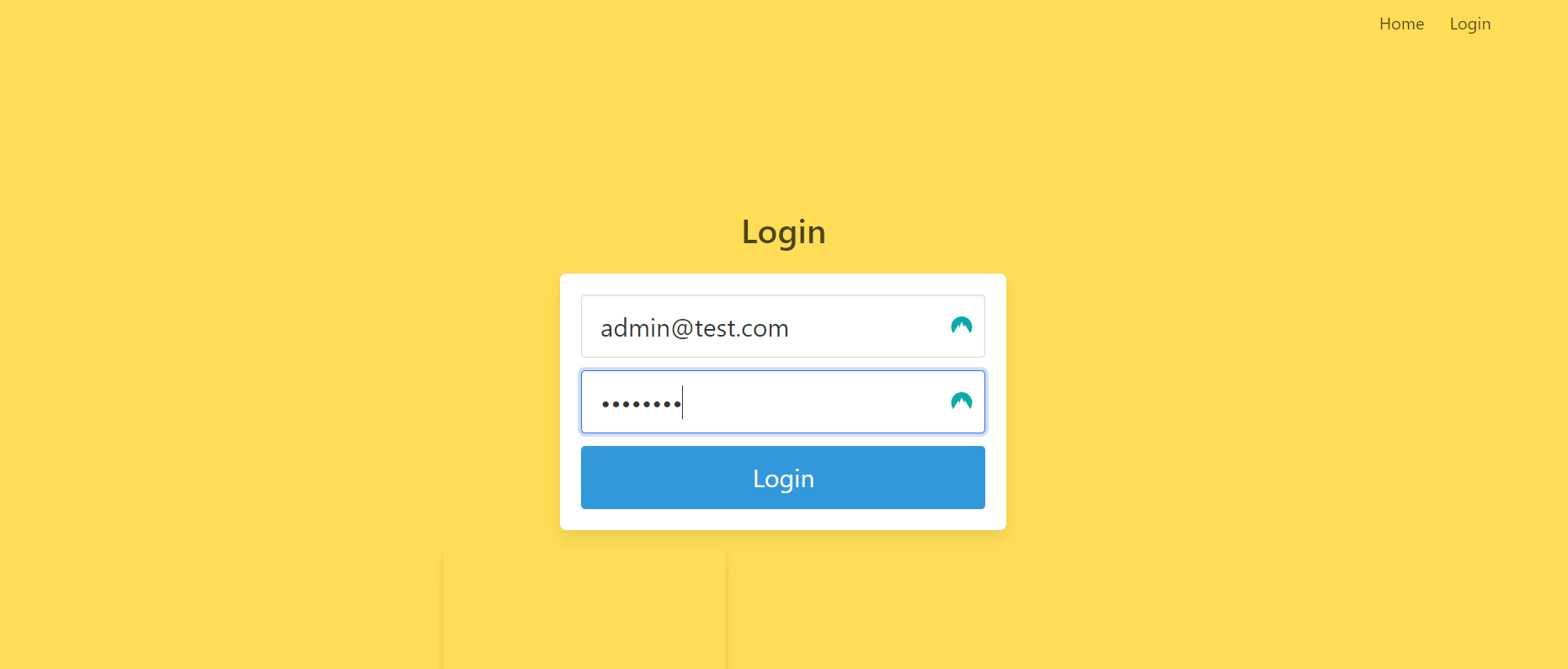


Figure 5: Login page

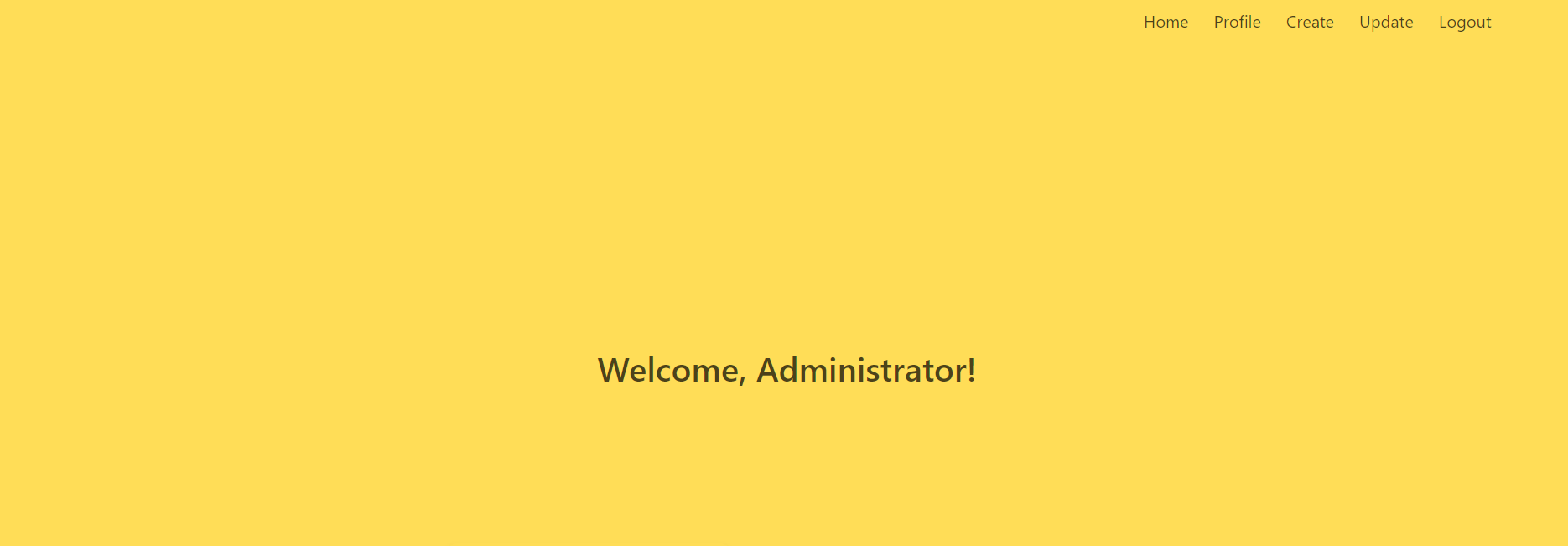


Figure 6: Welcome page

Create User

1. Select 'Create' in the top right of the screen and enter the new user's email, name, password, and role. Roles can be defined as:

'ISS'

'Ground staff'

'Government'

1. Click the blue 'Create' button.
2. Send QR code securely to user
3. Install FreeOTP app on user's device
   * Android: <https://play.google.com/store/apps/details?hl=en&id=org.fedorahosted.freeotp>
   * iOS: <https://apps.apple.com/us/app/freeotp-authenticator/id872559395>
4. Scan the generated QR code for OTP and two-factor authentication for the newly created user. Most recognised OTP apps are supported, however we recommend the use of FreeOTP:

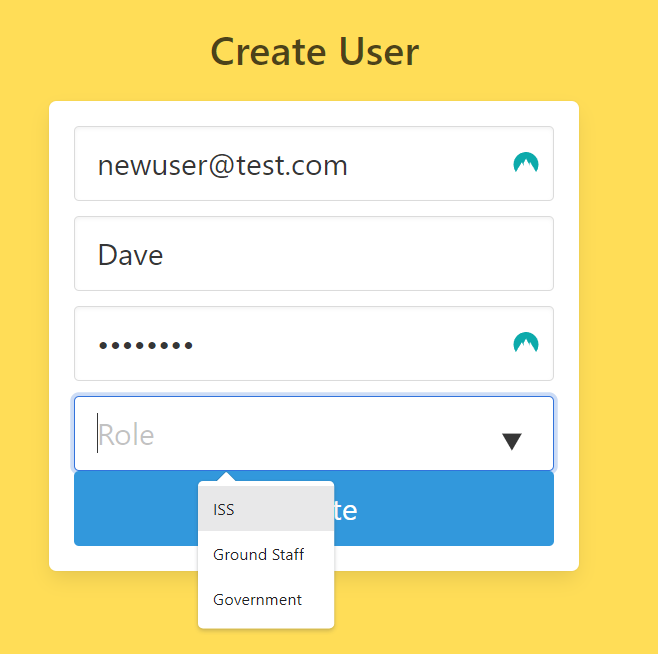


Figure 7: Create User page

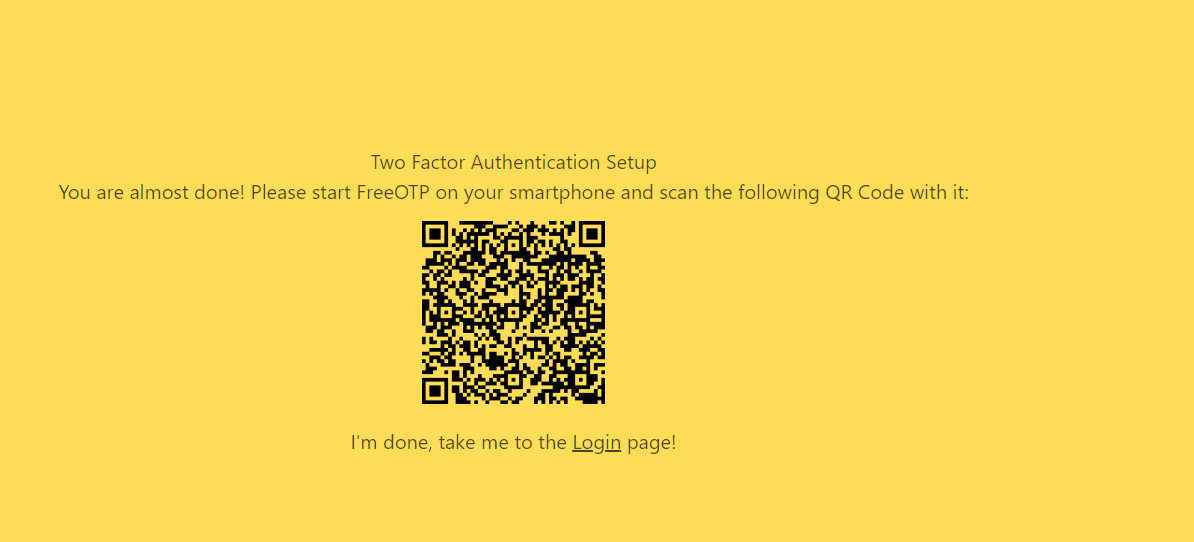
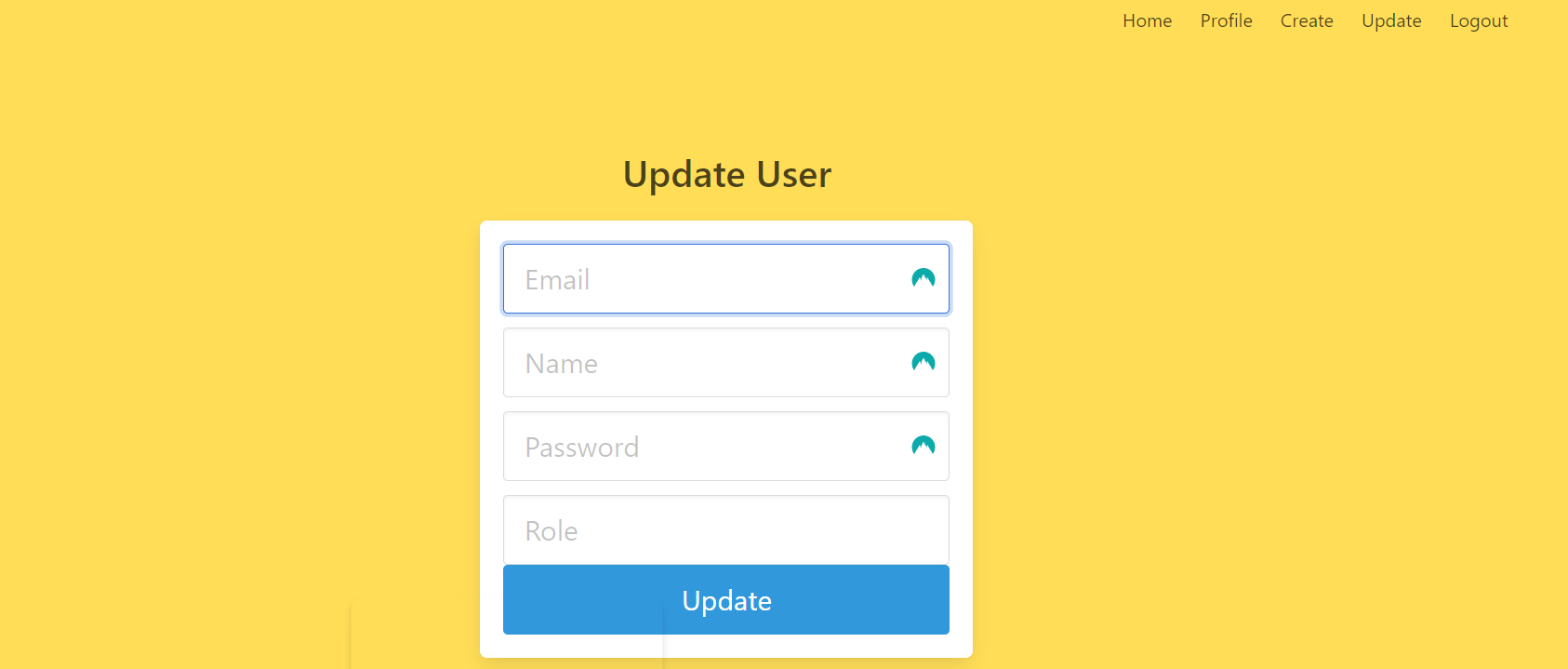


Figure 8: OTP initialisation

Update User

1. Click 'Update'.
2. Amend the necessary information
3. Click the blue 'Update' button.



Logout

1. Click 'Logout' from the user menu.

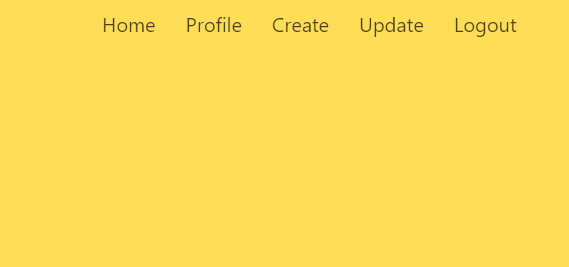


Figure 9: Admin menu

**User:**

Setup

Download the application from <https://github.com/ychan2020/SSD2022.git> and run it in the downloaded directory

1. In the CLI, cd [downloaded directory path]

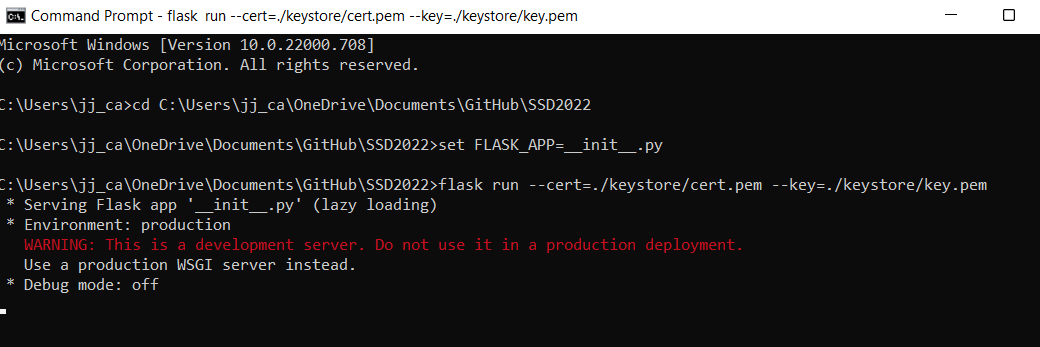


Figure 10: Running application

1. Set FLASK\_APP=\_init\_.py (Windows) or export FLASK\_APP=\_\_init\_\_.py (Linux)
2. flask run --cert=./keystore/cert.pem --key=./keystore/key.pem
3. Open the Client page <https://localhost:5000/>

Login

1. On the webpage, enter the email, password and OTP (FreeOTP).



Figure 11: Welcome page

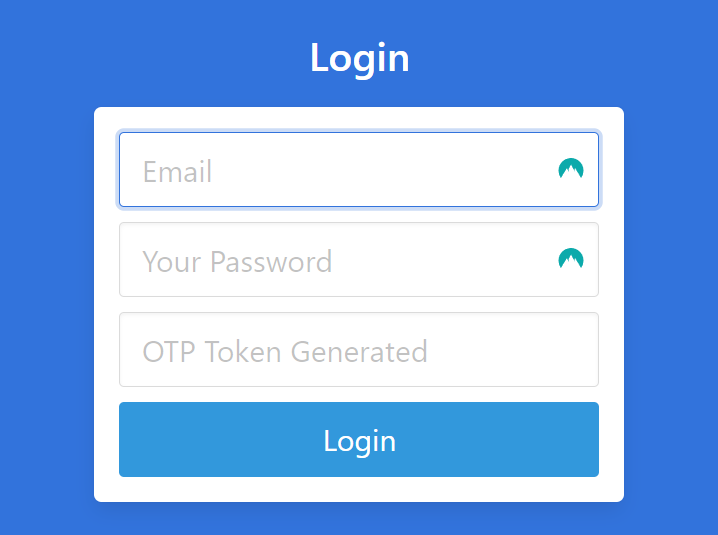


Figure 12: Login page

Change Password

1. Navigate to 'Profile'.
2. Enter a new password and click 'Change Password' (optional).

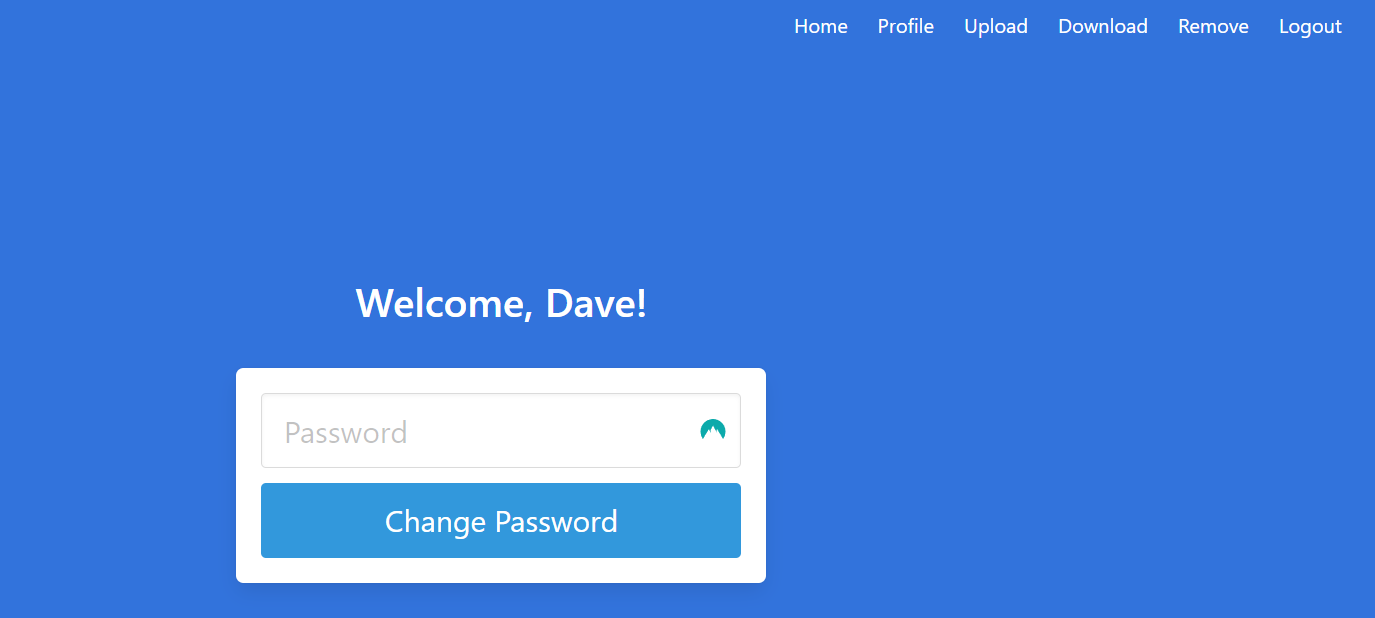


Figure 13: Change Password

Upload File

1. Navigate to 'Upload'.
2. Click 'Choose File' to be prompted to select a file to upload.
3. Click the blue button 'Upload'.

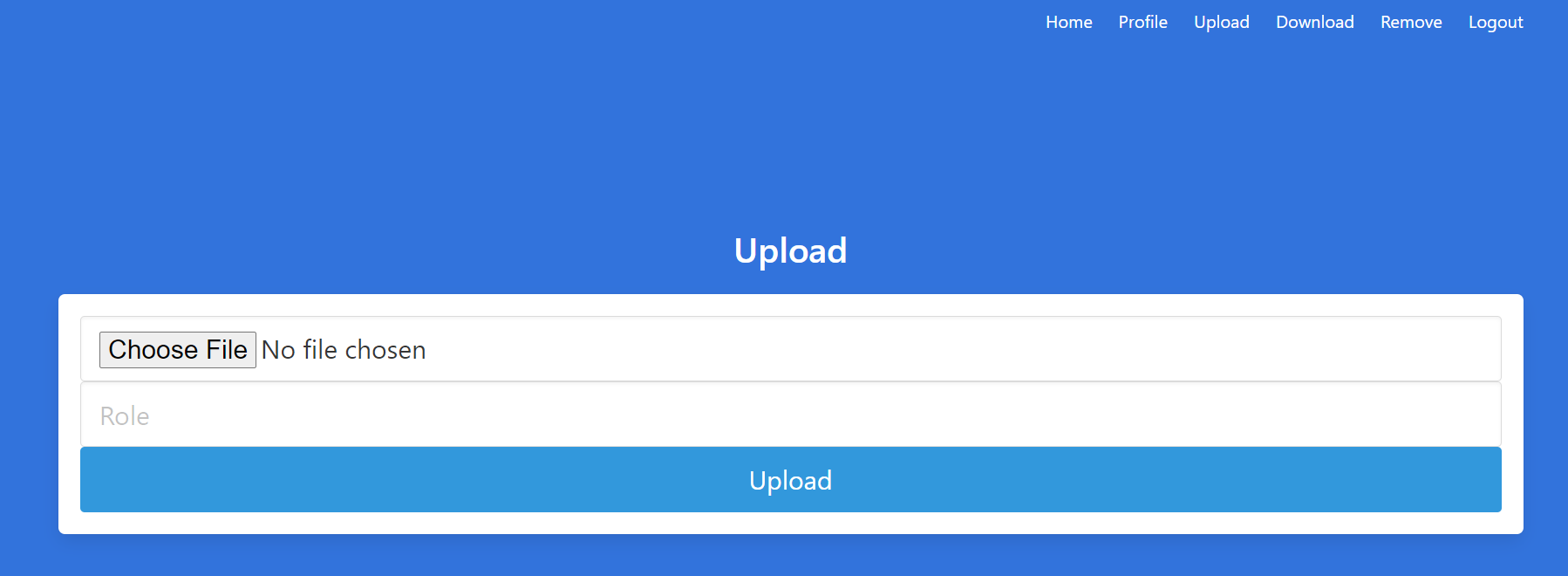


Figure 14: Upload File

Download File

1. Navigate to 'Download'.
2. View the list of files available due to privilege.
3. Select a File ID and click the blue button 'Download'.

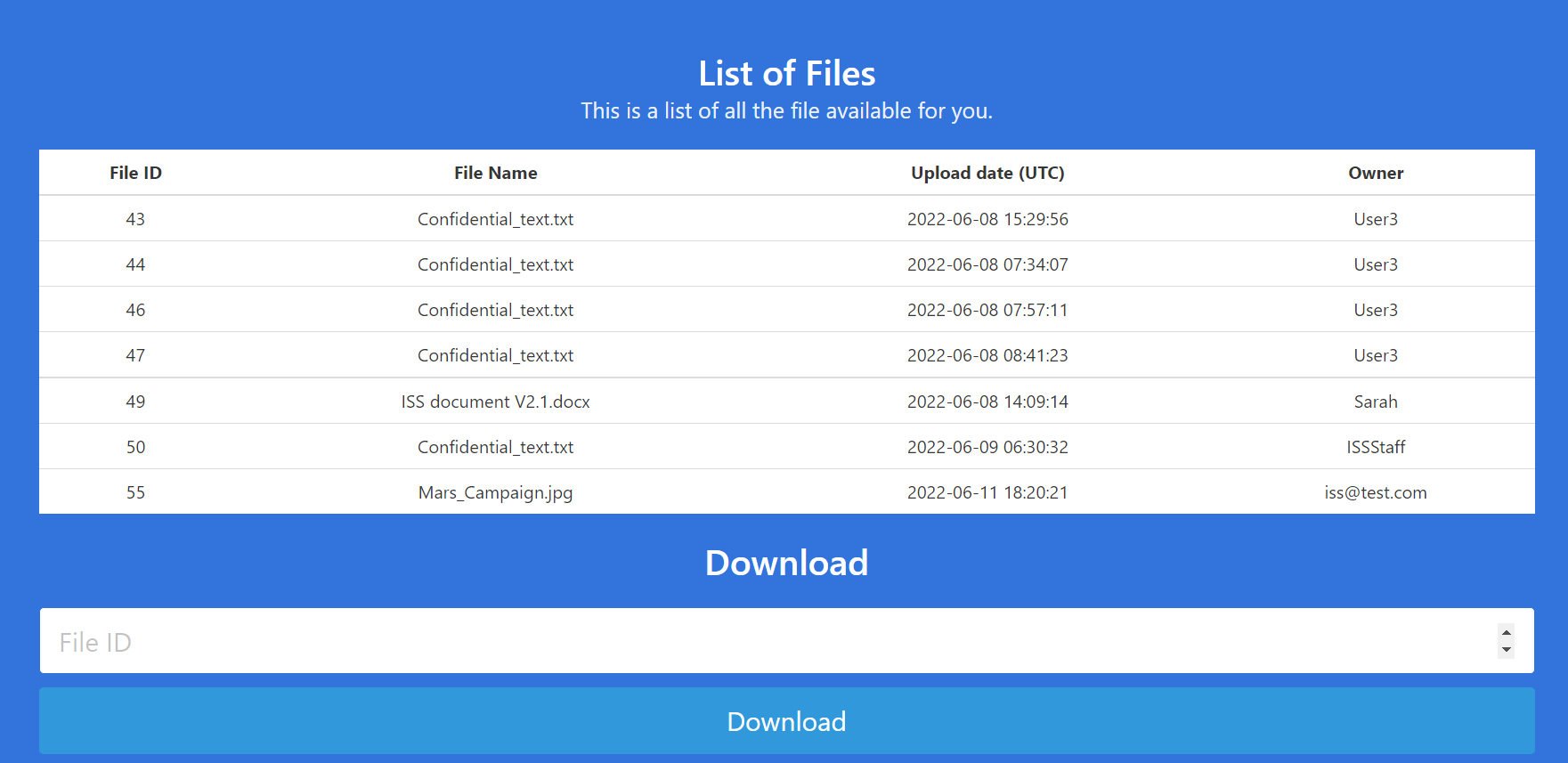


Figure 15: Download File

Remove a file

1. Navigate to 'Remove'.
2. View the list of files available due to privilege.
3. Select a File ID and click the blue button 'Remove'.

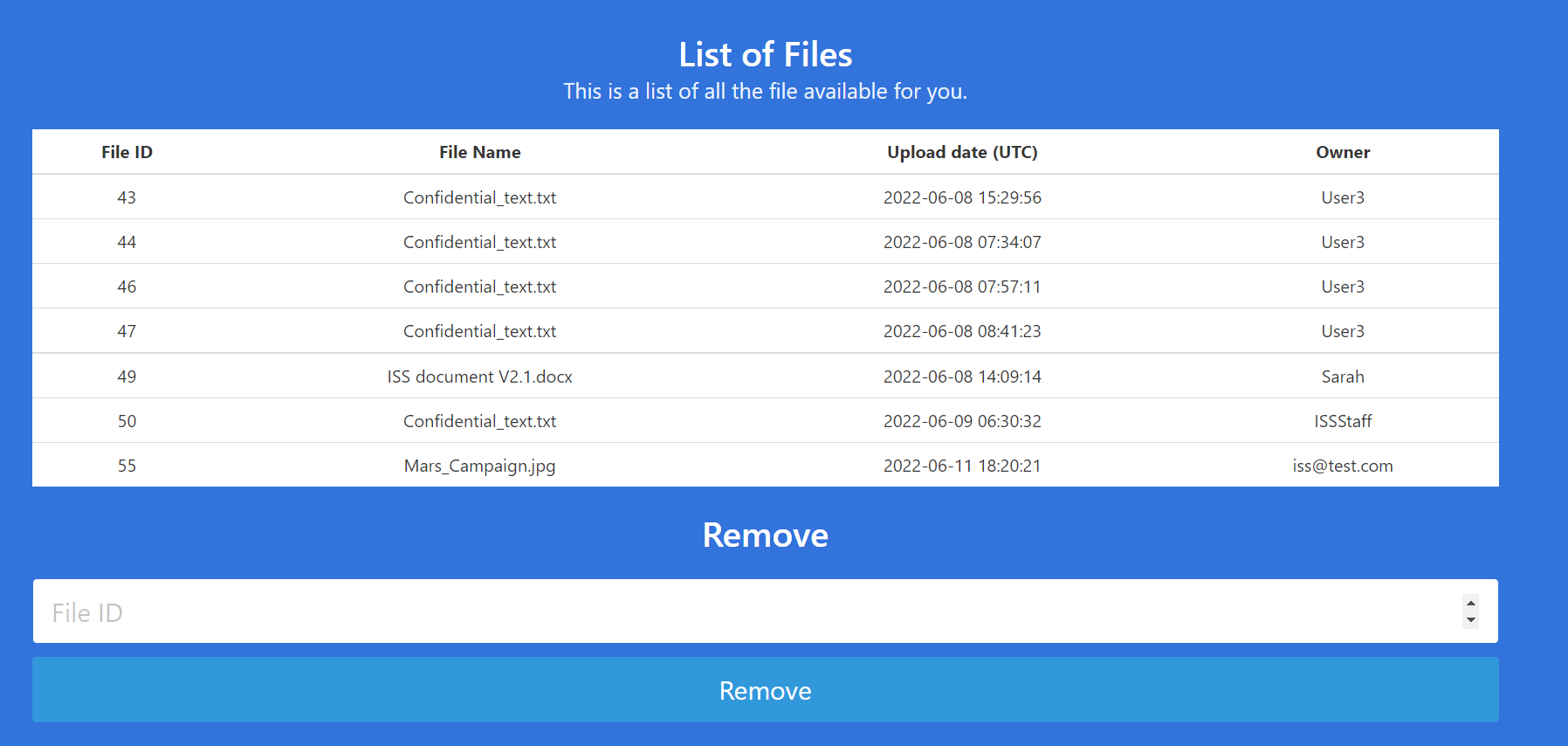


Figure 16: Remove File

Logout

1. Click 'Logout' from the user menu.

# Testing

Pylint and Flake8 testing for both user and admin interfaces was carried out at various stages of program development, and evidence can be found in Appendix B.

Code and functionality testing was also carried out, evidence of which can be found in Appendix C.

# References

Grinberg, M. (2015) Factor Authentication with Flask. [online] blog.miguelgrinberg.com. Available from: <https://blog.miguelgrinberg.com/post/two-factor-authentication-with-flask> [Accessed 02 June 2022]

phpMyAdmin. (N.D.) Available from: <https://www.phpmyadmin.co> [Accessed 06 June 2022].

Thales. (N.D.) One Time Password (OTP, TOTP) : definition, examples. Available from: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/technology/otp> [Accessed 12 June 2022].

The Pallets Projects. (N.D.) Flask. Available from: <https://palletsprojects.com/p/flask> [Accessed 08 June 2022].

‌

# Appendices

**APPENDIX A: Database SQL**

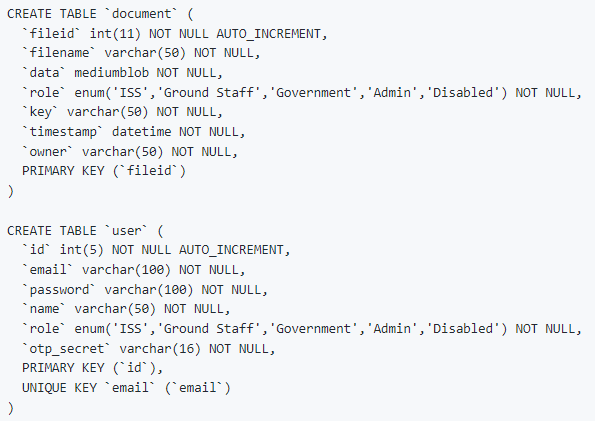


Figure 17: Database Creation SQL Query

**APPENDIX B: Pylint and Flake8 Testing**

User interface

07 June 2022:

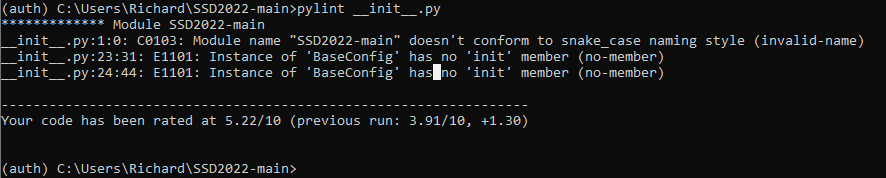


Figure 18: \_\_init\_\_.py

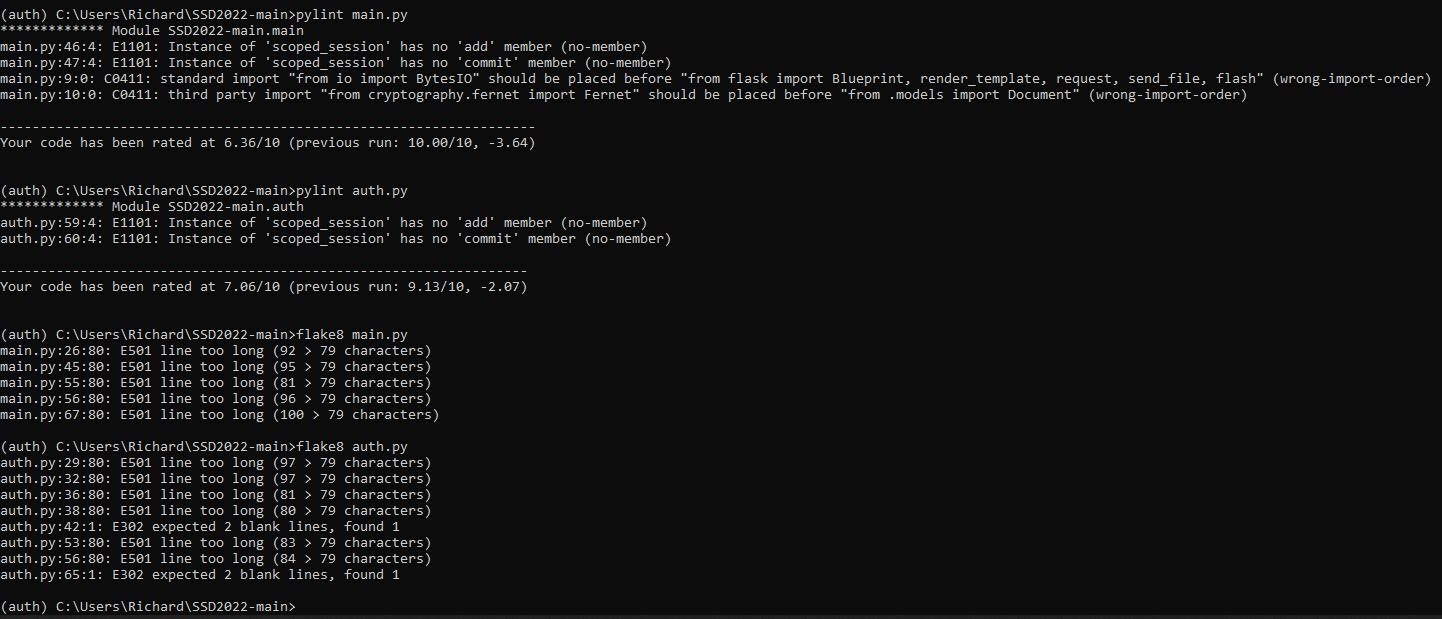


Figure 19: auth.py and main.py

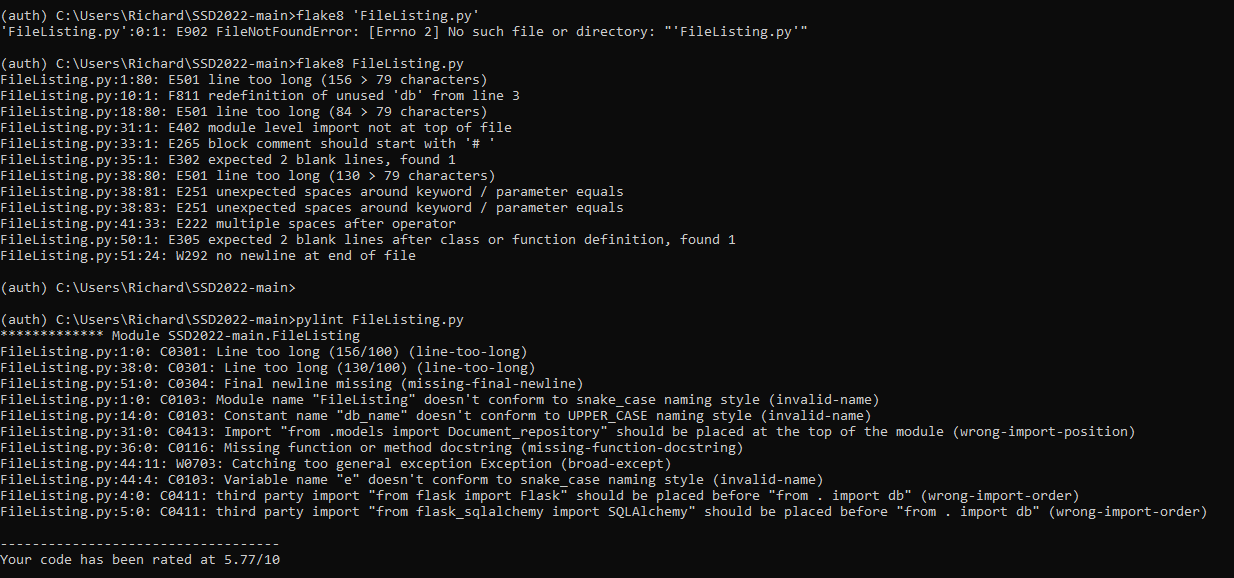


Figure 20: File Listing.py (prototype)

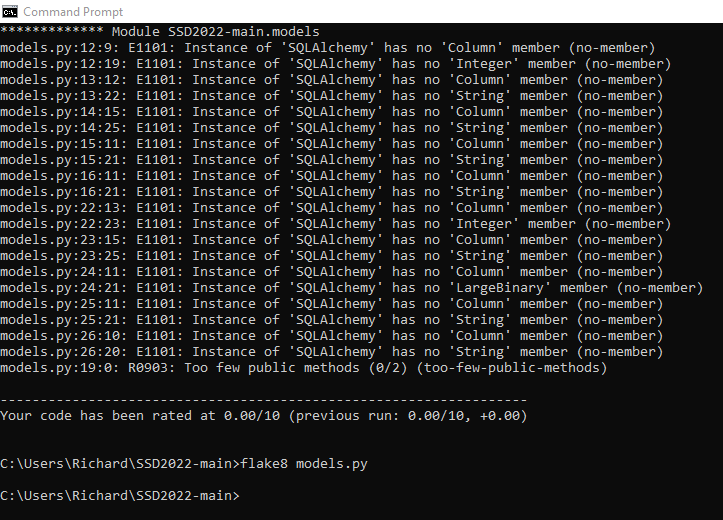


Figure 21: models.py

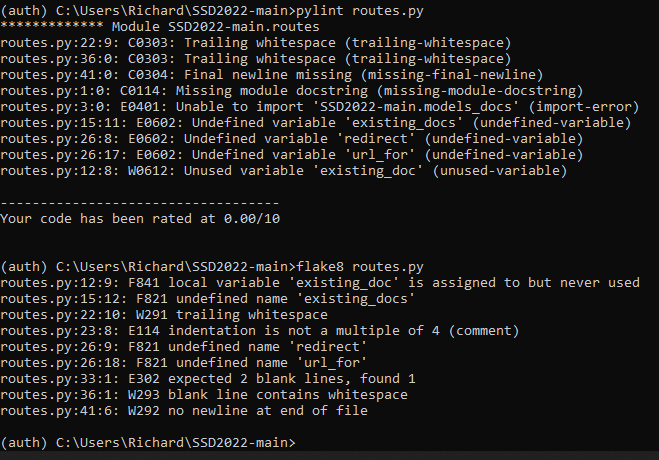


Figure 22: routes.py (not used in the final program)

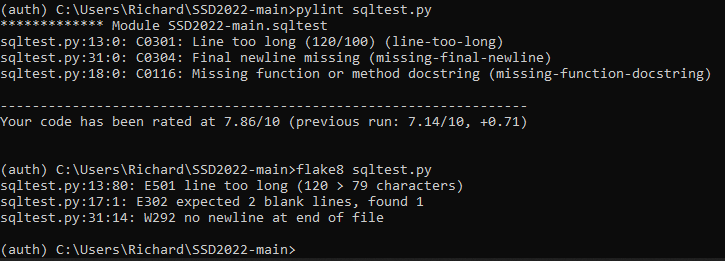


Figure 23: sqltest.py (DB connection test file)

10 June 2022:

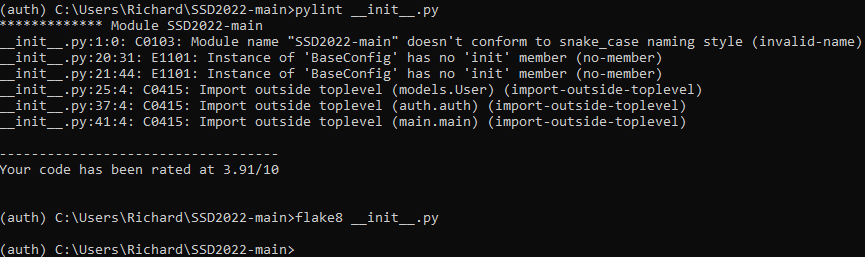


Figure 24: \_\_init\_\_.py

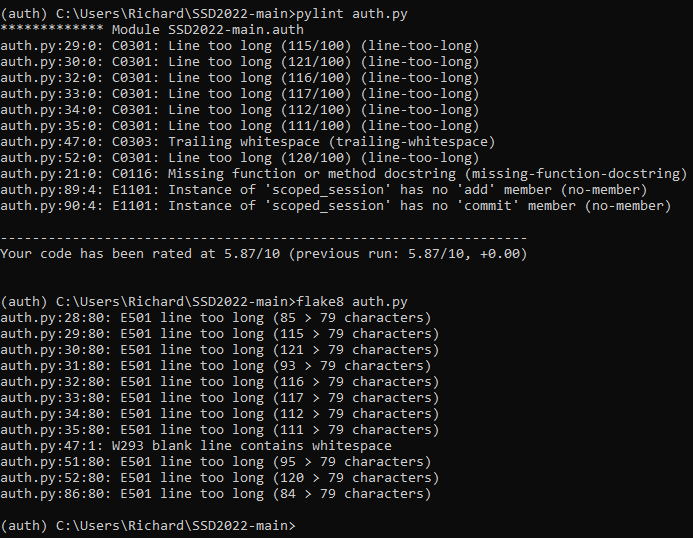


Figure 25: auth.py

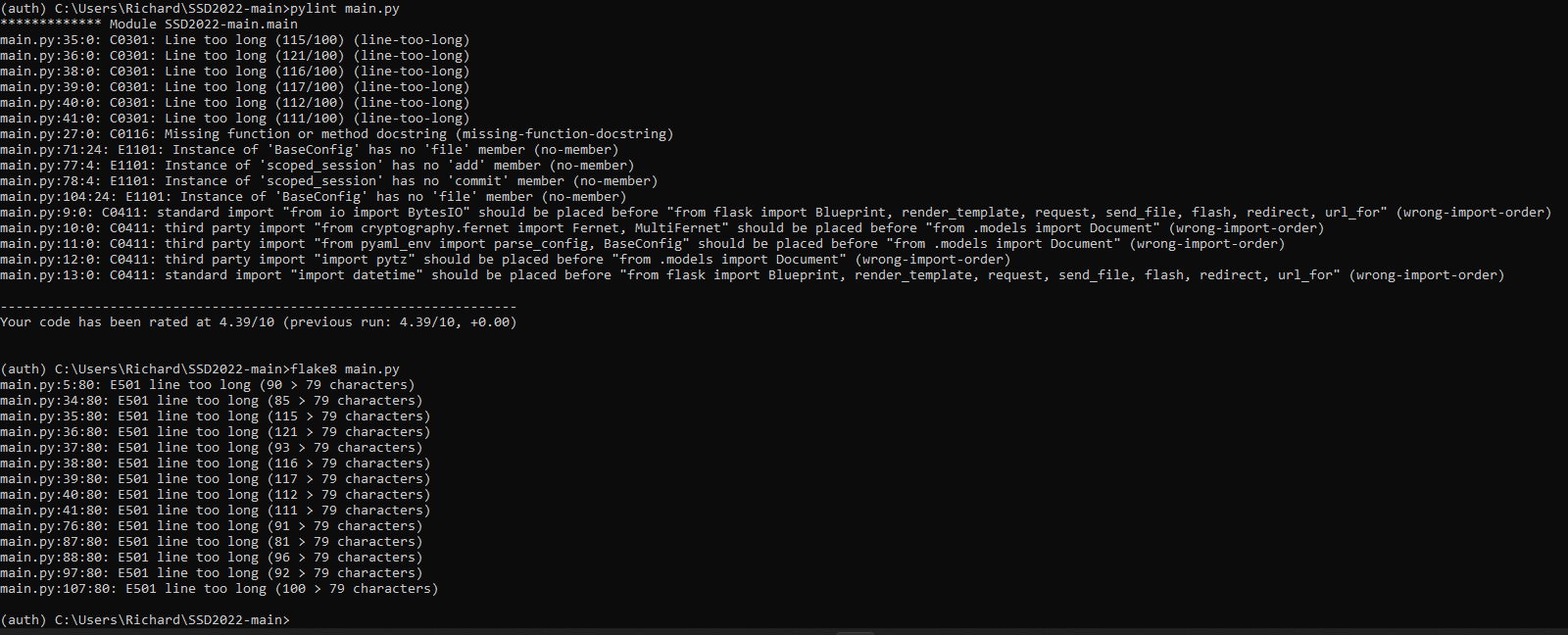


Figure 26: main.py

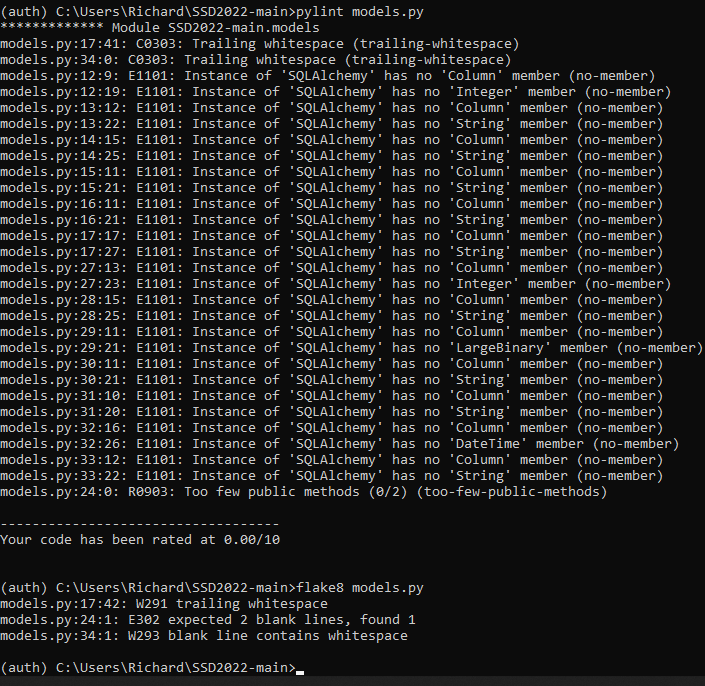


Figure 27: models.py

13 June 2022:

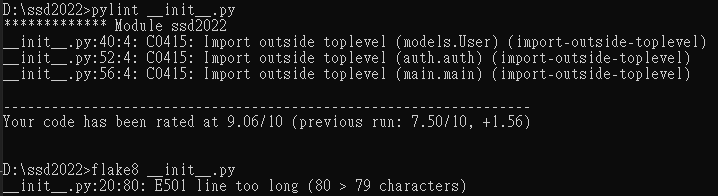


Figure 28: \_\_init\_\_.py

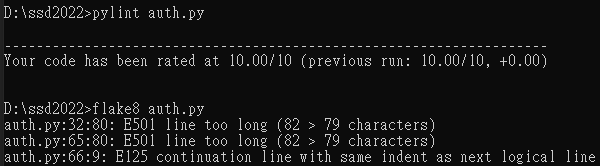


Figure 29: auth.py

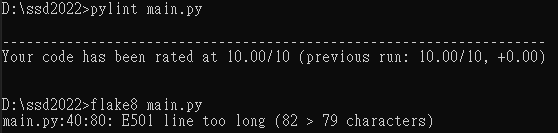


Figure 30: main.py

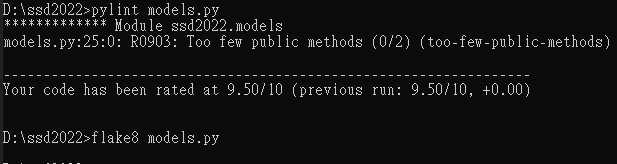


Figure 31: models.py

Admin interface

08 June 2022:

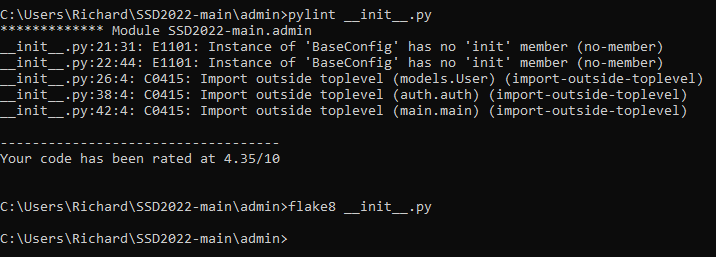


Figure 32: \_\_init\_\_.py

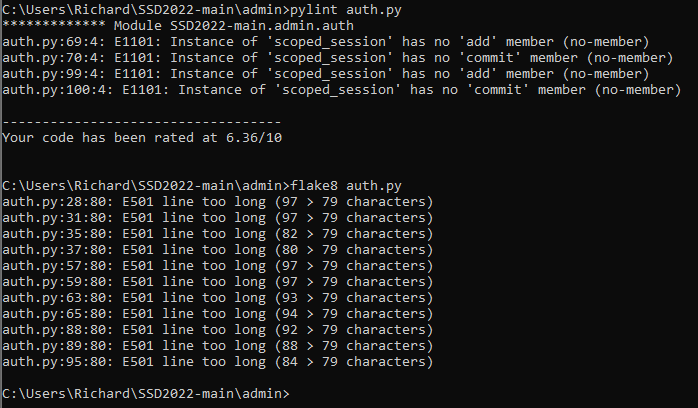


Figure 33: auth.py

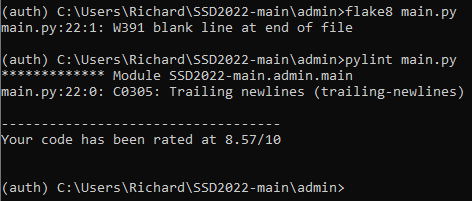


Figure 34: main.py

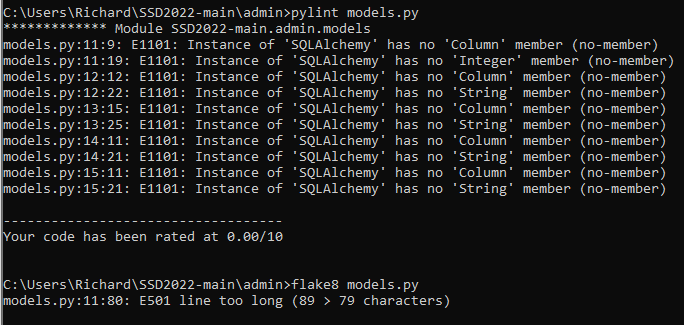


Figure 35: models.py

10 June 2022:

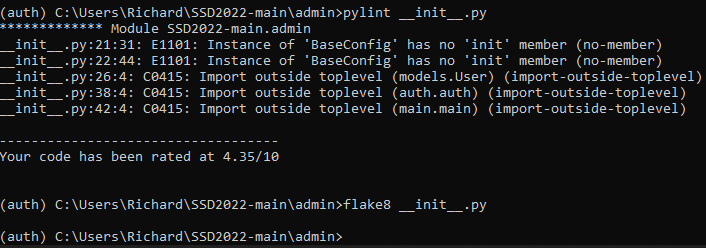


Figure 36: \_\_init\_\_.py

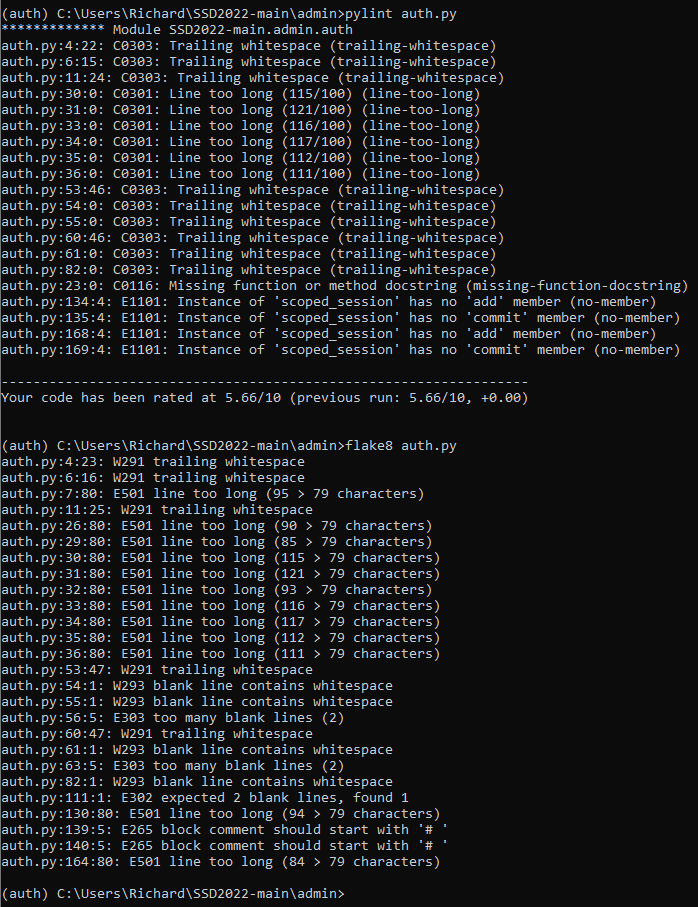


Figure 37: auth.py

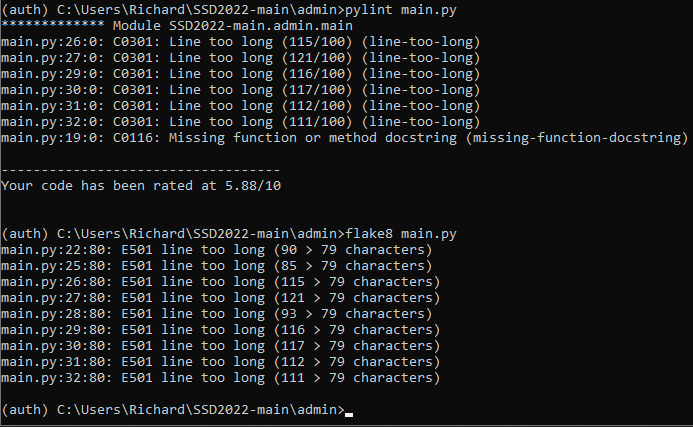


Figure 38: main.py

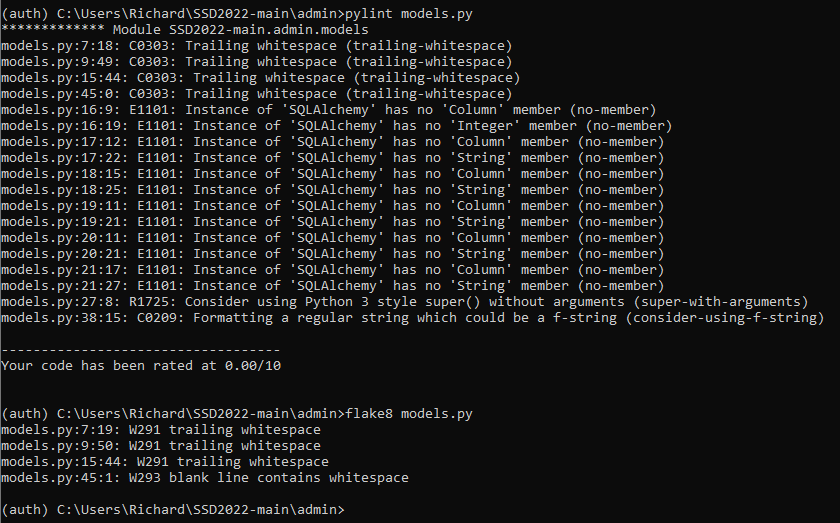


Figure 39: models.py

13 June 2022:

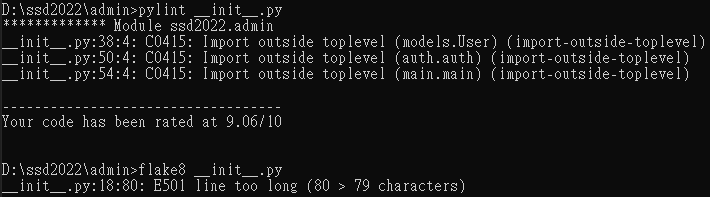


Figure 40: \_\_init\_\_.py

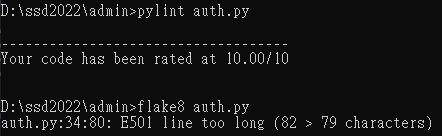


Figure 41: auth.py

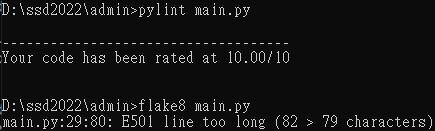


Figure 42: main.py

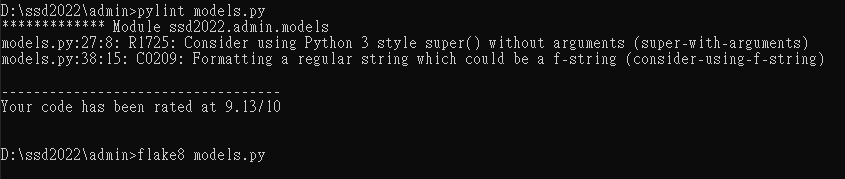


Figure 43: models.py

**APPENDIX C: Functional and Code Testing**

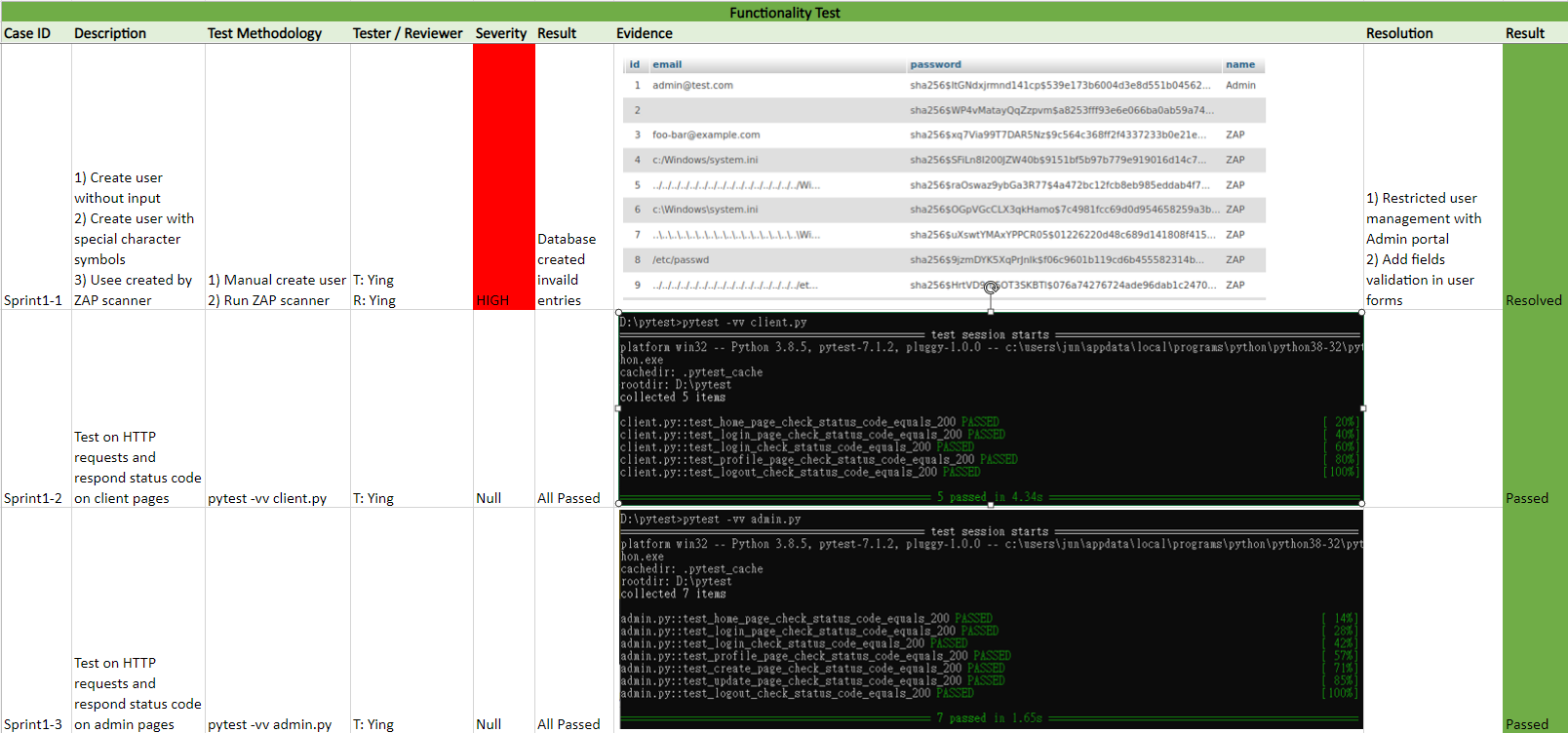


Figure 44: Sprint 1 - Functional Testing

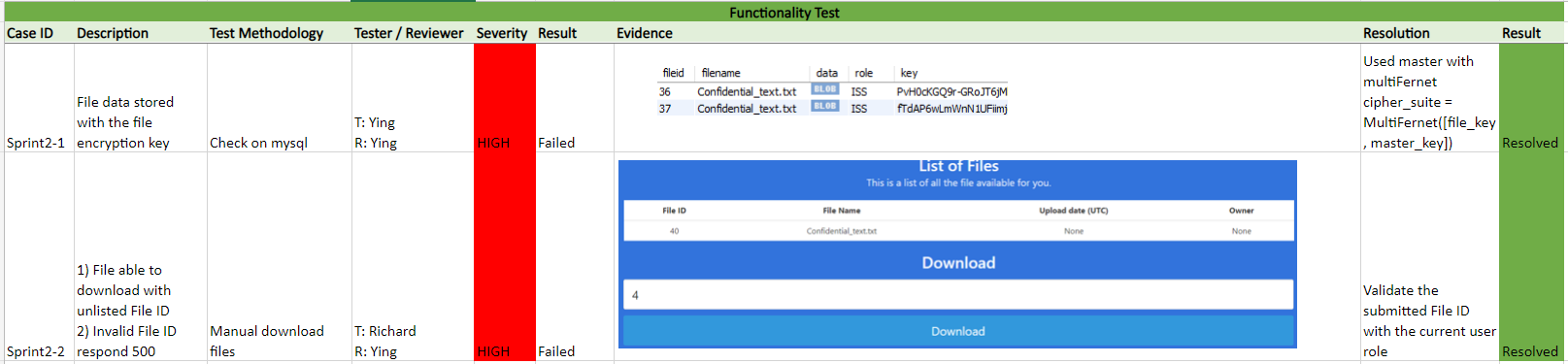


Figure 45: Sprint 2 - Functional Testing

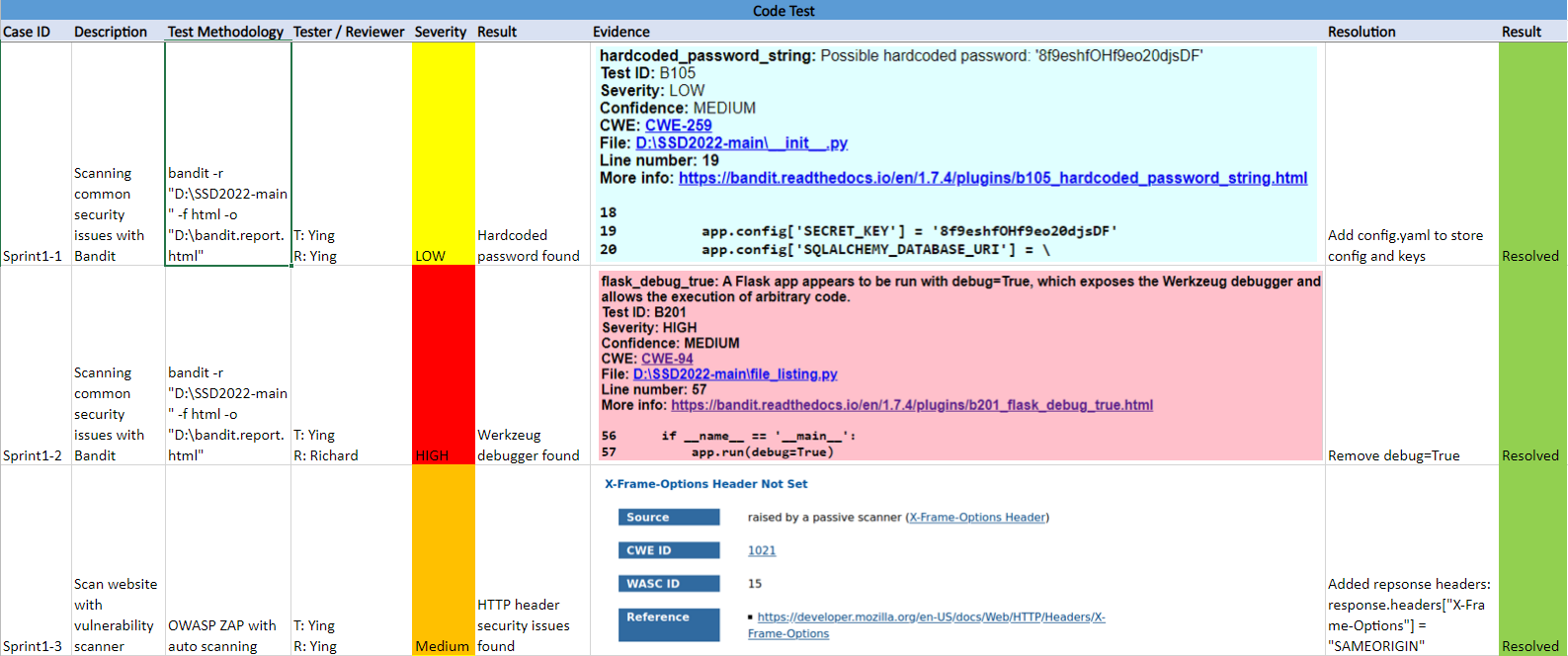


Figure 46: Sprint 1 - Code Testing

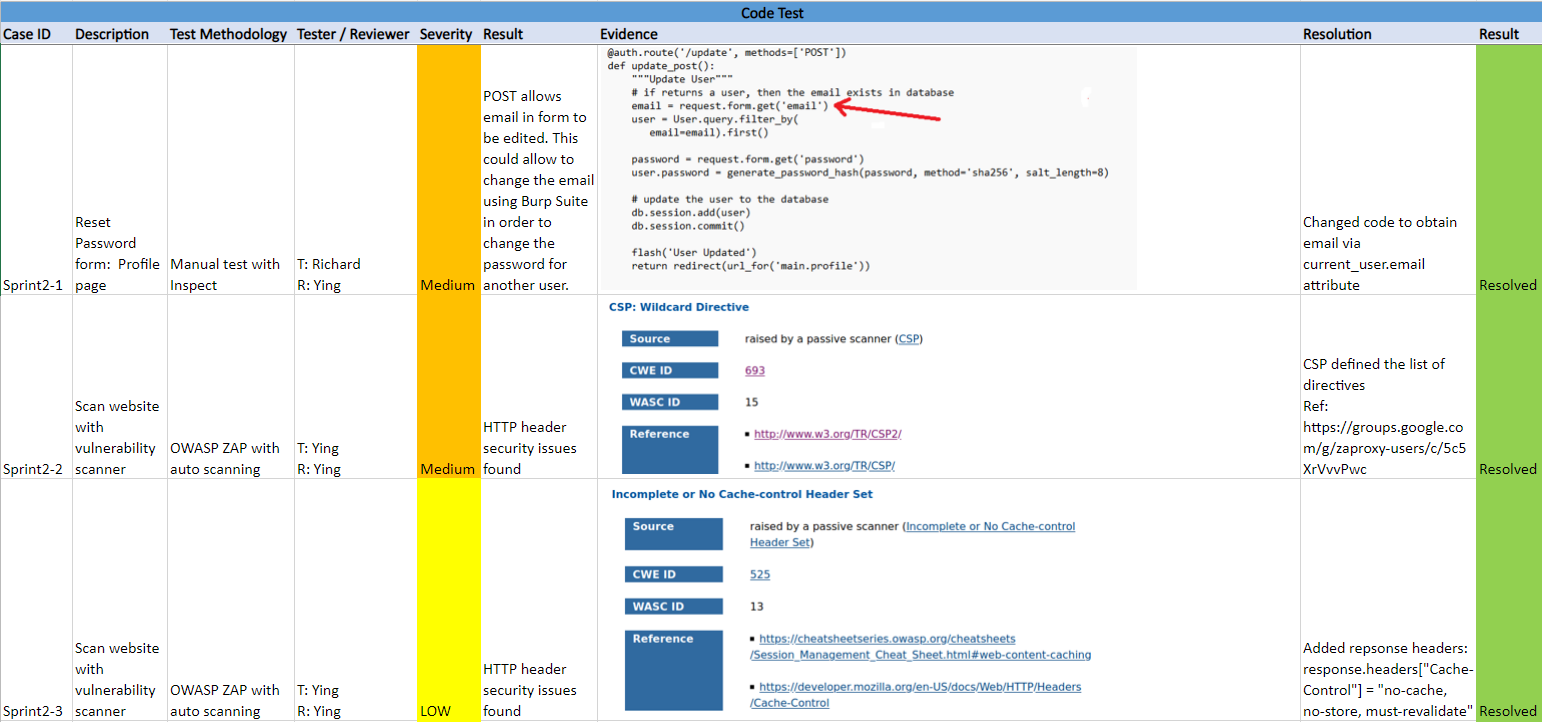


Figure 47: Sprint 2(a) - Code Testing

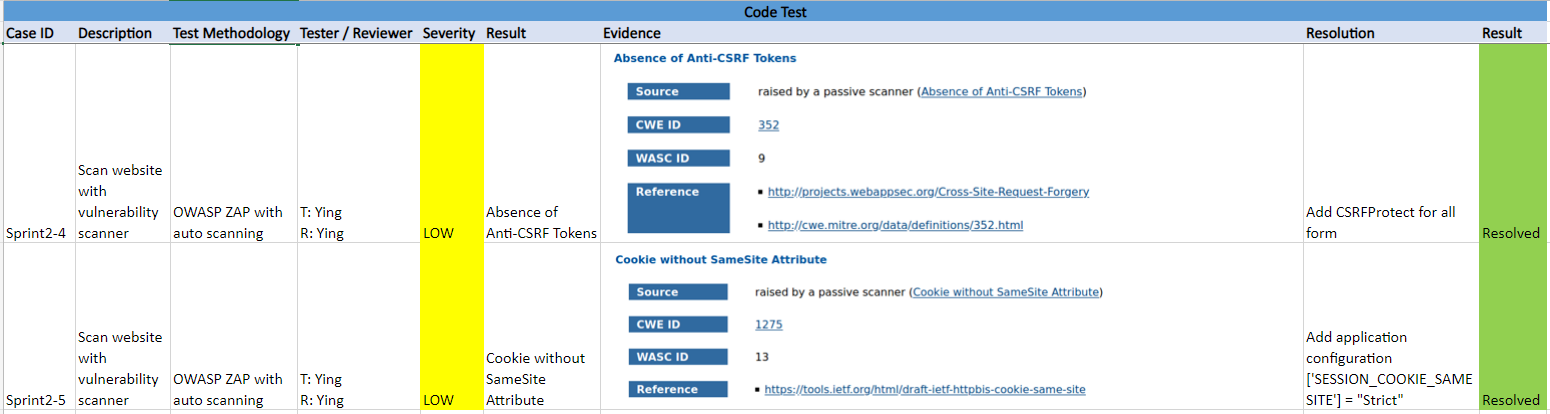


Figure 48: Sprint 2(b) - Code Testing