**Development Team Project**

*International Space Station: Testing Document*

MSc Cyber Security, Secure Software Design, Unit 6 Submission

[**Functional Testing**](#_u8ikrbw3u6b6) **2**

[Strong Password Entry](#_ybx38x21qun9) 2

[Legitimate Email Address](#_vggy1exrw6z3) 2

[Correct Re-Entry of Password](#_v1d4ouxix9ub) 3

[Multiple Incorrect Entries](#_5mb9i3wxlrtw) 3

[Successful Registration](#_fq9fbwxbjbir) 4

[Unregistered User attempting Login](#_dqyeve39tj46) 4

[Incorrect password](#_jfhugm18709p) 5

[Successful Login](#_p1x83sqayk0e) 5

[Level 0 Access](#_rgnnatupyyou) 6

[Level 2 Access](#_ia57xlujj2ze) 6

[Admin Level Access](#_i4zudgukkshk) 7

[Uploading a file](#_72onn9nmo6vw) 7

[Uploading a Confidential file](#_a0eeoutlh64p) 7

[Uploading a potentially malicious file](#_6a1ewun28krl) 8

[Updating Metadata](#_u6iuzgsf2a4v) 9

[Files Encrypted](#_4xdjcomopohd) 9

[Level 0 Show Files](#_9q1xrigsb1uj) 10

[Level 2 Show Files](#_6rywow7zbtge) 10

[Admin Show Files](#_x2jht5vqt6s9) 11

[User Download of data](#_10ooevyymc2r) 11

[Deletion of files](#_vyc24itma8te) 12

[**Non-Functional Testing**](#_js335p1gfeig) **12**

[Unauthorised Access to data](#_mmkos5sqgrg3) 13

[Manage File Access / File Storage](#_obsafhff6pb2) 13

[Python Linter](#_9ic9bxk22rpr) 13

[Static Application Security Testing (SAST)](#_n2iwv4ya88fr) 15

# Functional Testing

This section covers functional testing, testing key operational requirements for the solution.

## Strong Password Entry

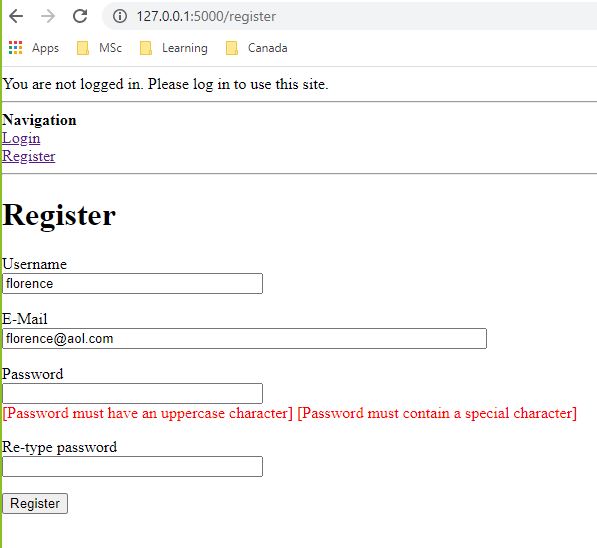
To ensure security to the highest level, the first test was to identify if a weak password could be used when registering for an account.

Test data

Username: florence

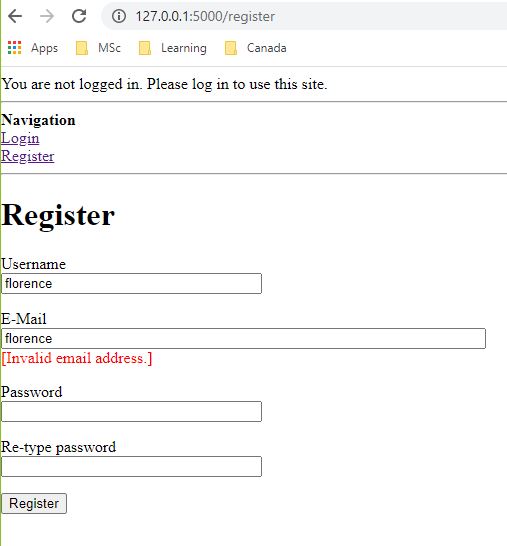
Email: [florence@aol.com](mailto:florence@aol.com)

Password: password123

Re-type: password123  


## Legitimate Email Address

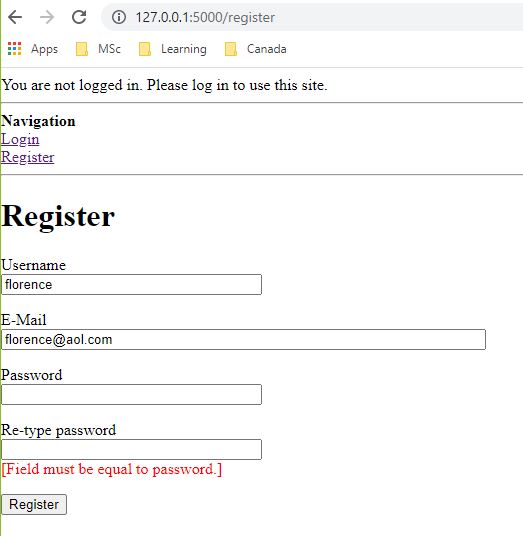
The next test was to see if an account could be registered which did not have a legitimate email address.

Test data  
Username: florence  
Email: florence  
Password: Password!23  
Re-type: Password!23  


## Correct Re-Entry of Password

Next the correct re-entry of password was tested, again when registering as a user.

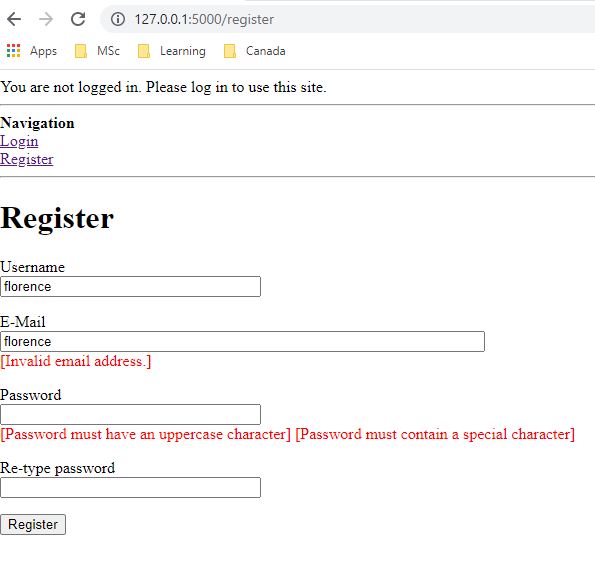
Test data  
Username: florence  
Email: florence@aol.com  
Password: Password!23  
Re-type: Password!24



## Multiple Incorrect Entries

A test was also conducted to show if multiple errors would come up on the screen, if details were typed incorrectly.

Test data  
Username: florence  
Email: florence  
Password: password123  
Re-type: password123



## Successful Registration

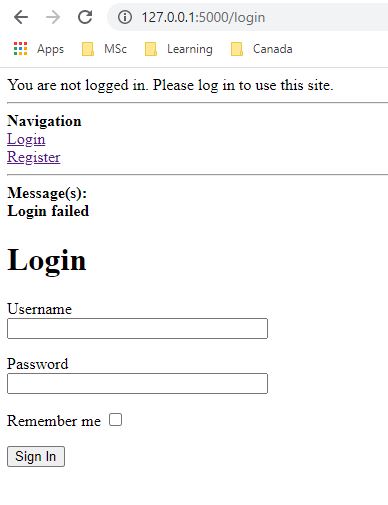
Finally, a test to show the successful registration of a new user.

Test data  
Username: florence  
Email: florence@aol.com  
Password: Password!23  
Re-type: Password!23



## Unregistered User attempting Login

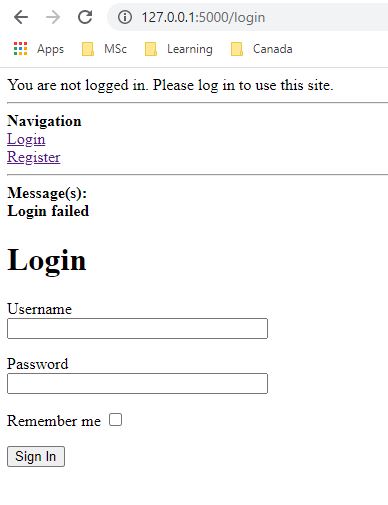
This test was focused on the user login, specifically an unregistered user attempting to login.

Test data  
Username: george  
Password: Password!23  


## Incorrect password

This test showed what happened when a registered user typed in an incorrect password.

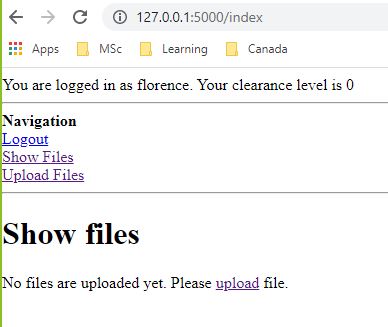
Test data  
Username: florence  
Password: Password!24



## Successful Login

This screenshot shows the successful login.

Test data  
Username: florence  
Password: Password!23



When testing the login, the screenshots show that the error message is the same for when you attempt to login without registering and when you are a registered user but have entered the incorrect password. This was chosen to ensure the message was ambiguous to any potential malicious actors. Whilst conducting the login tests, logout was also tested and found to be successful.

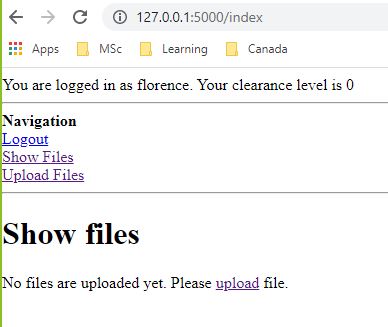
The next test was focused on User Role Assignments: level 0, level 2 and Admin.

Test data

| **Username** | **password** | **Clearance Level** |
| --- | --- | --- |
| keyser | QW12$#as | Administrator |
| lisa | AS13#@zx | 2 |
| florence | Password!23 | 0 |

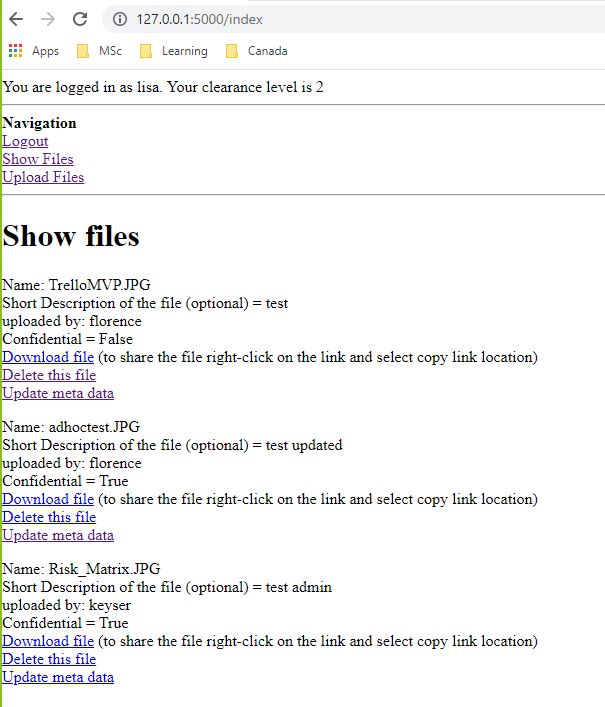
## Level 0 Access

This screenshot shows the access of a Level 0 user.



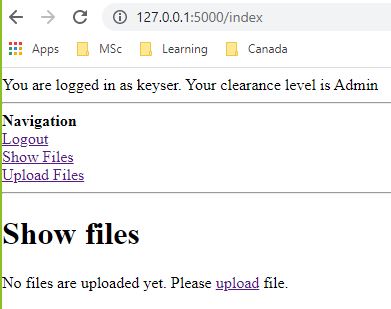
## Level 2 Access

This screenshot shows the access of a Level 2 user.



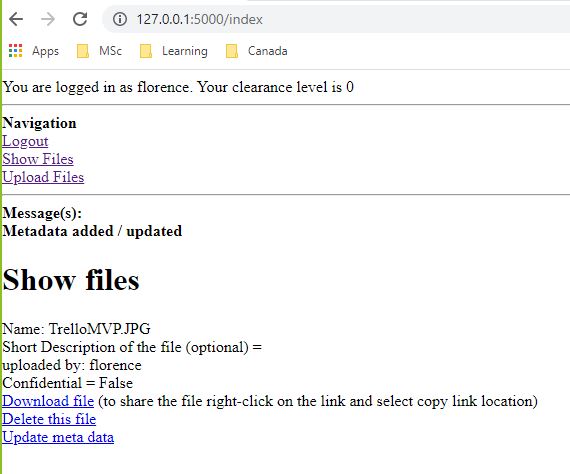
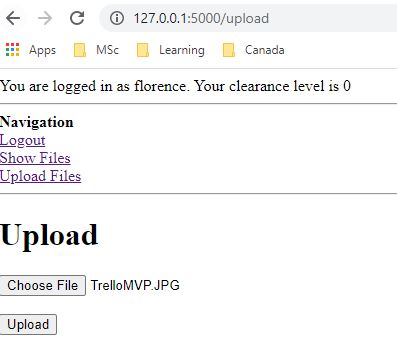
## Admin Level Access

This screenshot shows the access of an Admin user.



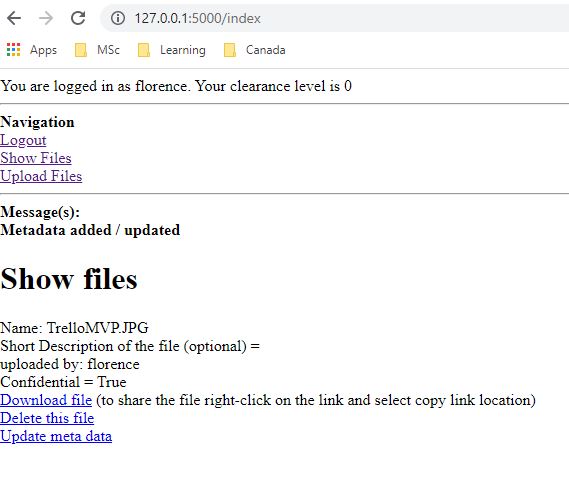
## Uploading a file

This screenshot and the one below it show the process of uploading a file.



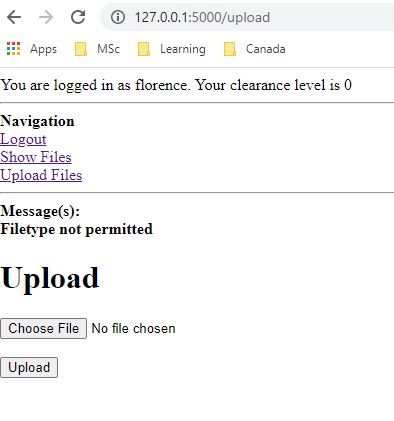
## Uploading a Confidential file

This screenshot shows the uploading of a confidential file.



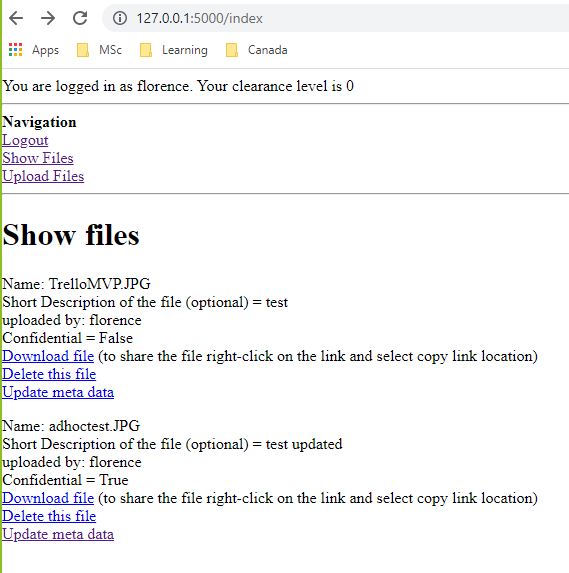
## Uploading a potentially malicious file

This screenshot shows the error message received when attempting to upload a file type that has not been allowed.



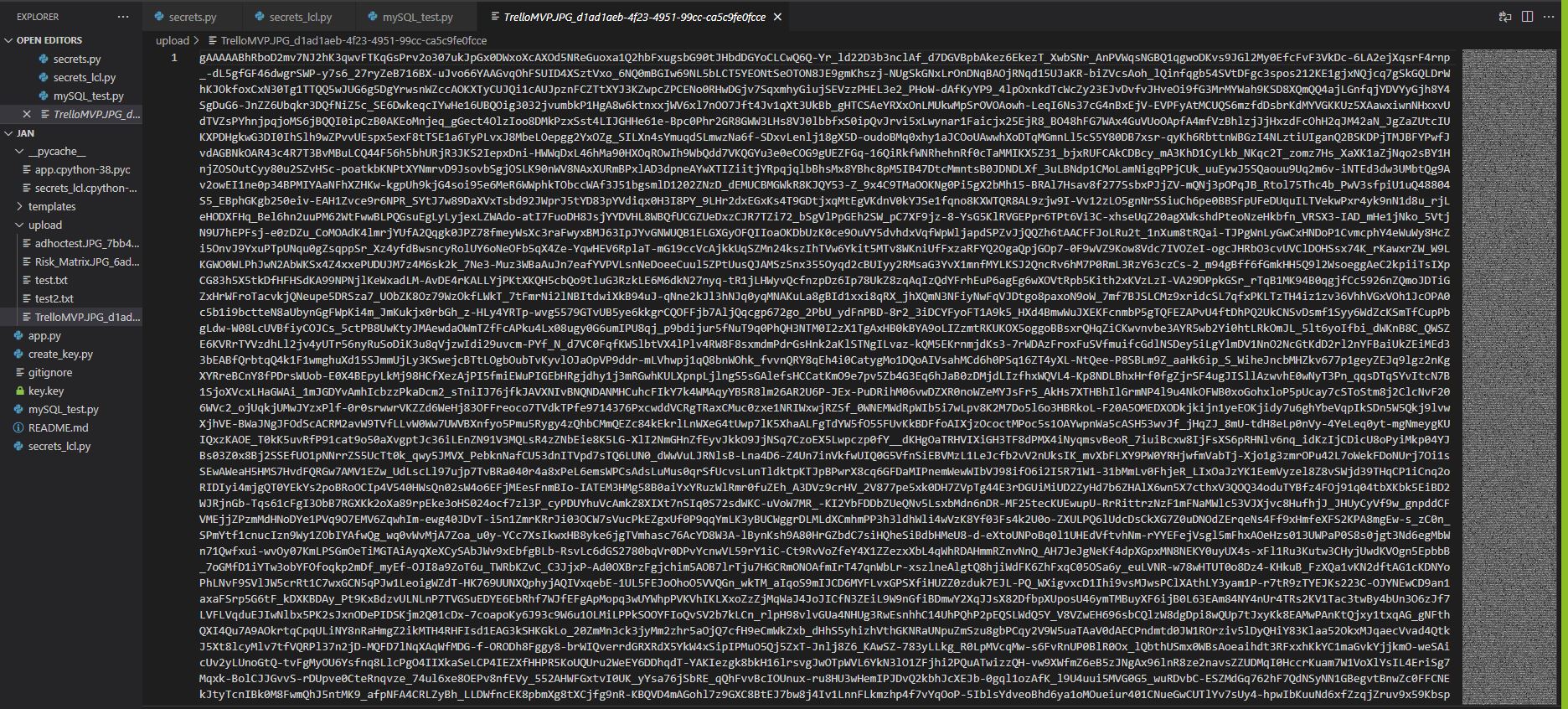
## Updating Metadata

This screenshot shows an update to the metadata to the file called TrelloMVP.JPG.



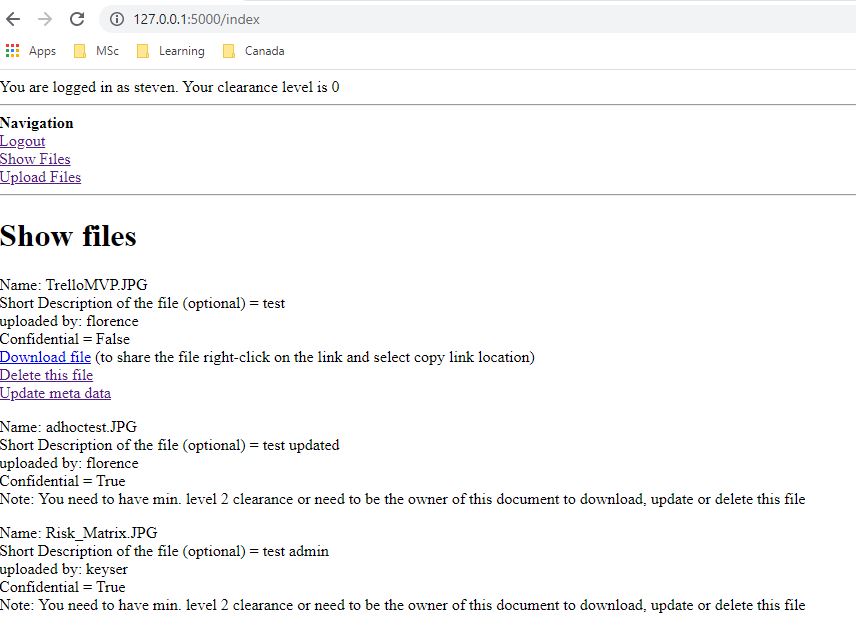
## Files Encrypted

This screenshot shows one of the files uploaded via Visual Studio. You can see that this has been encrypted and cannot be read without decryption.



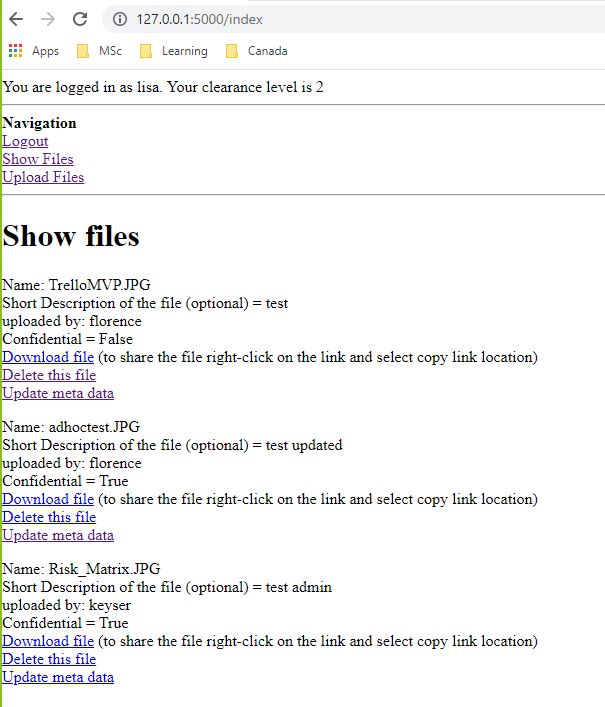
## Level 0 Show Files

This screenshot shows the files a Level 0 can see. They cannot download, delete or update the metadata of any confidential documents.



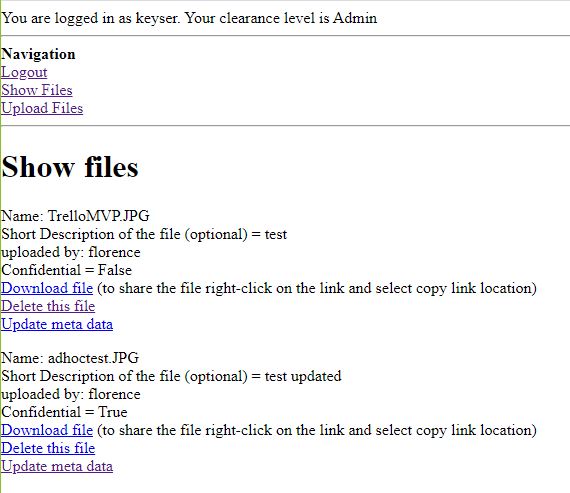
## Level 2 Show Files

This screenshot shows the files a Level 2 can see.



## Admin Show Files

This screenshot shows the files an Admin can see.



Something to consider would be the restriction of the admin role further as ideally we would like it that not one user would be able to do everything, so this needs to be looked at during further stages of development.

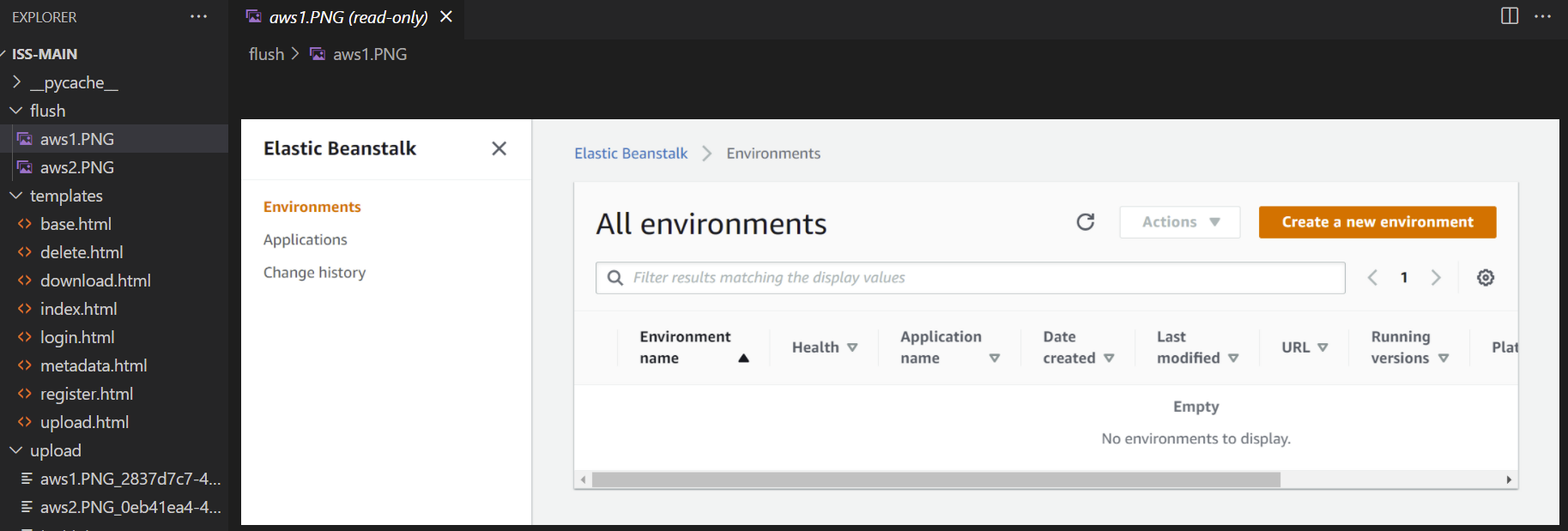
As an area for development, Level 2 and Admin access are currently the same. These will need to change in future developments.

## User Download of data

Screenshots below show the process of downloading a file. File is saved in the flush directory of the project as a decrypted file.

## 

## 



## Deletion of files

When uploading data, the deletion of files was also tested and found to work successfully.

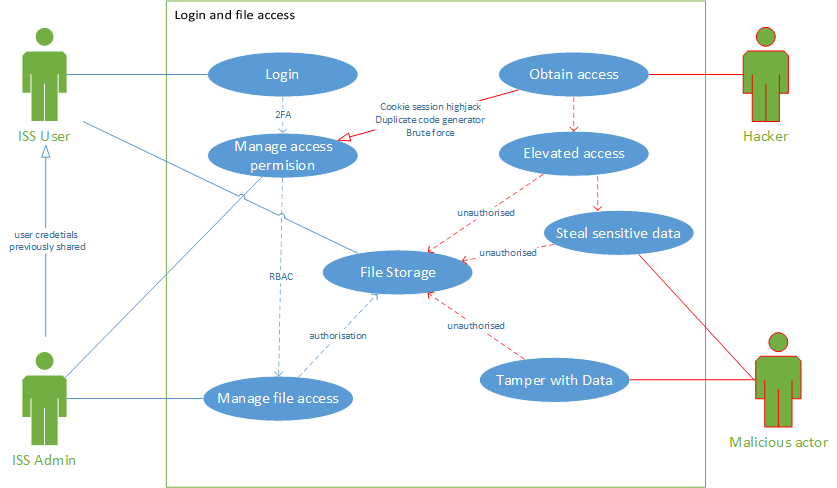
# Non-Functional Testing

**Vulnerability Testing based on Abuse Case diagram**

Tool: Manual

Rationale: Abuse cases are born out of considering the function and capabilities of software from the perspective of an attacker (Damodaran, 2006). Using these to inform testing ensures coverage of a wide range of threats.

Abuse Case Diagram:



## Unauthorised Access to data

To resolve the potential threat of unauthorised access to data, it has been found that the system created ensures there are different role levels. Having different roles and access mitigates the potential risk of unauthorised access; however, it does not necessarily resolve the issue of human error, e.g. if an internal user accidentally provides access to an unauthorised user. This is mitigated with further controls.

## Manage File Access / File Storage

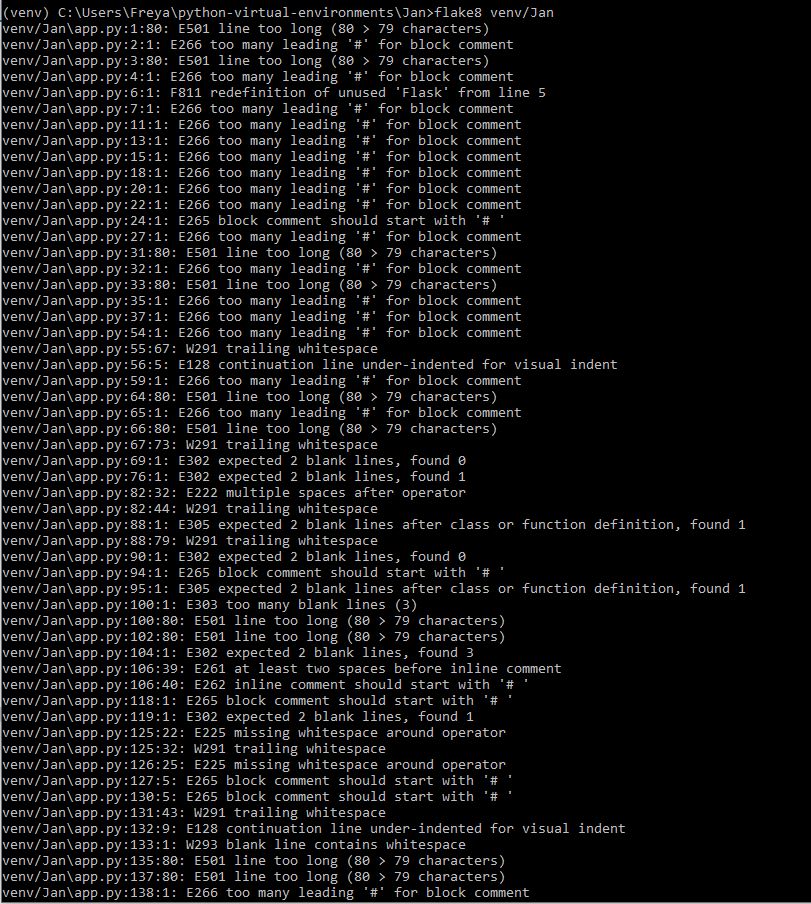
As seen previously, the role the user is assigned changes their access to the system e.g. a level 0 role cannot download, update or delete confidential information.

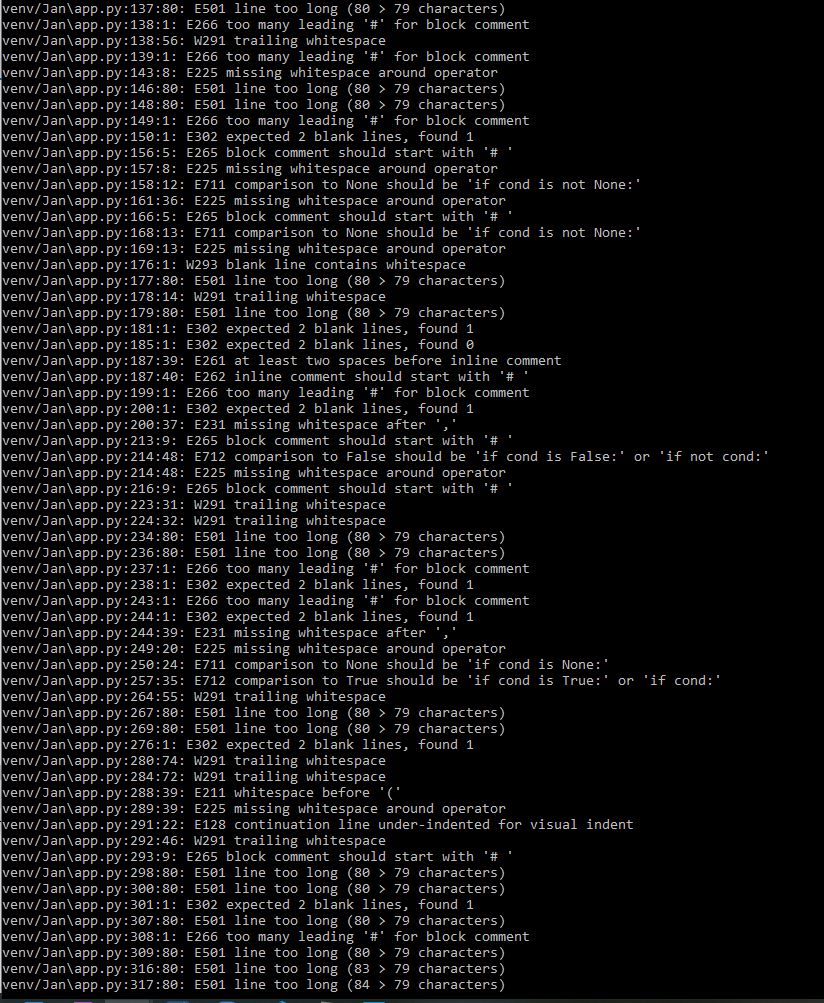
## Python Linter

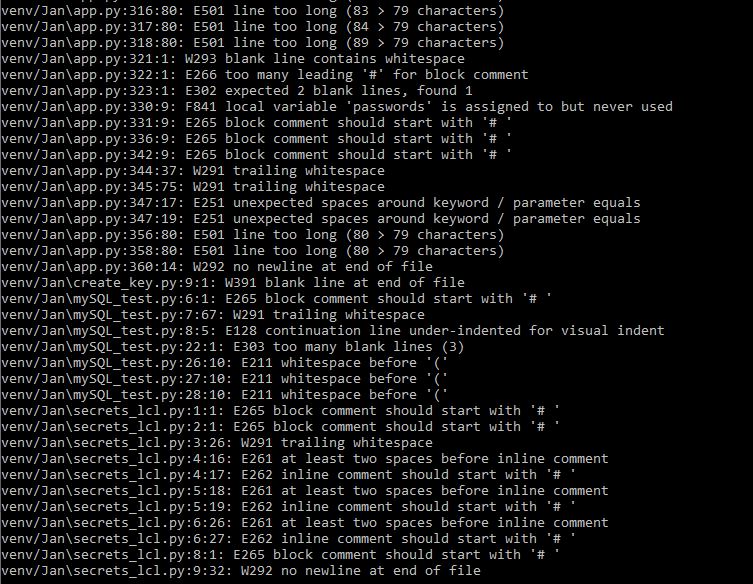
Tool: Flake8

Rationale: Linters test for adherence to best practice, for example PEP-8 for Python. Flake8 has been chosen as it has more features than other Linters such as PyLint, including testing for cyclomatic complexity (Flake8, 2021).

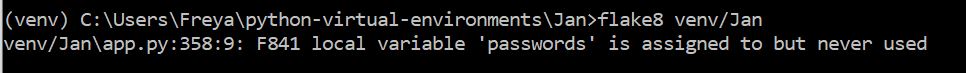
The below screenshots show the outcome of running the Flake8 tool. This testing has been provided back to the developers so changes can be made to the code.







Update: Changes were made to the code and Flake8 ran again with the following finding which has since been resolved.

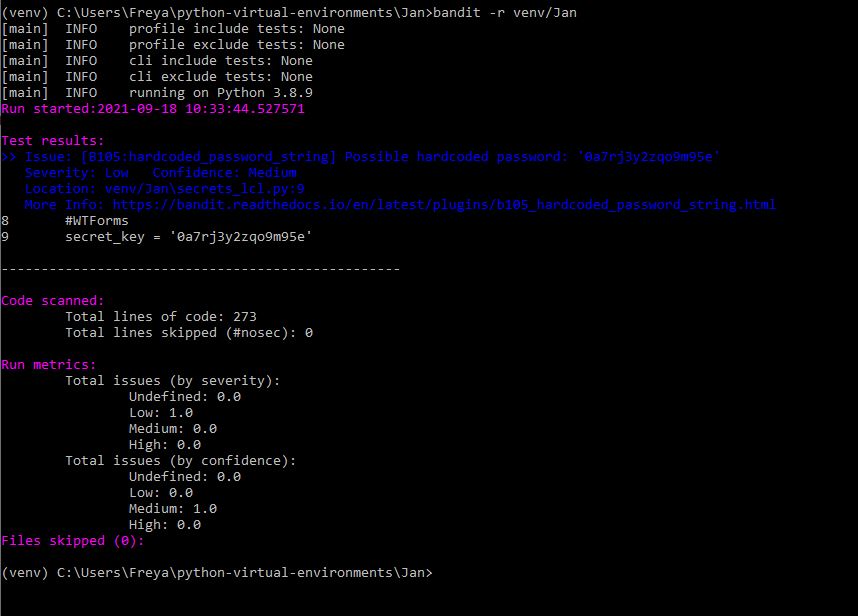


## Static Application Security Testing (SAST)

Tool: Bandit

Rationale: SAST tools identify common security flaws in code. LGTM appears to be more comprehensive and supported, however it requires the GitHub repository to be public and this is not the case for this application (LGTM, N.D.). Bandit will be used instead as it can be used locally, is specific to Python and is free (OWASP, 2021).

The screenshot below shows the outcome of the Bandit scan.



From this scan, it was found that the secret key is hardcoded. This is something that typically is suggested against and should be changed in future versions of this development (Beyond Trust, 2019).

References

Damodaran, M. (2006) Secure Software Development Using Use Cases and Misuse Cases. *Issues in Information Systems* 7(1): 150-154.

Flake8. (2021) Flake8: Your Tool For Style Guide Enforcement. Available from: <https://aws.amazon.com/compute/sla/> [Accessed 18 September 2021].

LGTM (N.D.) About LGTM. Available from: <https://lgtm.com/help/lgtm/about-lgtm> [Accessed 18 September 2021].

OWASP (2021) Source Code Analysis Tools. Available from: <https://owasp.org/www-community/Source_Code_Analysis_Tools> [Accessed 18 September 2021].

Beyond Trust (2019), Hardcoded and Embedded Credentials are an IT Security Hazard - here's what you need to know. Available from: <https://www.beyondtrust.com/blog/entry/hardcoded-and-embedded-credentials-are-an-it-security-hazard-heres-what-you-need-to-know> [Accessed 19 September 2021]