Assignment 2

Development Team Project: Coding Output

Submission checklist:

- You should submit a fully tested set of code with all required libraries. Code should be well documented with in-line commentary.
- You should supply evidence of execution, demonstrating how key aspects of your code work (via demos, screenshots and output captures).
- You should include comprehensive evidence of testing including output from test tools (such as linters, etc.) demonstrating correct code, security and functional testing with any remediation activity documented as well.
- You should submit a README file that documents how the application runs
- Discussion of any differences between the design and the final code produced (600 words). This should be part of the README file.

Submission Date: 14 February 2022
Tutor: Dr Cathryn Peoples

Submitted by: Team Builder (Thien Liu & Neelam Pirbhai-Jetha)

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1. A brief overview

Digitpol, an international Internet Forensics and cybersecurity company in the Netherlands, provides businesses with extensive experience in cybersecurity and cybercrime investigations (DIGITPOL, 2021). Since security challenges (digital asset thefts and lack of protection of forensics data) and malevolent attacks and tampering of databases (Chopade & Pachghare, 2019) are numerous, our aim is to review Digitpol's repository system of all criminal files/documents so that it follows the security best practices (OWASP, 2021; ISO/IEC 27000:2018).

The main objective is to develop an application that provides a secure repository for Digipol that will allow authorised stakeholders to securely manage their documents in a protected repository which respects data privacy. The three aspects of software architecture – confidentiality, integrity, availability (CIA) – aided by authentication, authorisation, and non-reputability will be applied (Pillai, 2017).

1.2. Planning the development of the application

The agile framework (Leach, 2016) is usually used for team work. It has been decided to use the Kanban framework to plan the work to be done:

PLANNED	IN PROGRESS (BUILD AND TEST)	DONE
Brainstorm the different elements of the app		
Create the app using Flask step by step		
Checking security vulnerabilities (see class notes)		
Testing of app (see class notes)		

Table 1: Work Plan

The Development Design Document submitted in Unit 6 will be followed, and the choices made during the development phase will be discussed in each section below.

2. README FILE

2.1. Title

Digitpol Repository

2.2. Description

A secure repository based on a hierarchical authentication model, is being designed for DIGITPOL. The Access Control policy is crucial to control users' permissions and will protect against "unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits" (OWASP, 2021).

2.3. Development Tools and Libraries:

- Programming Language: Python 3.x
- Source Code Version Control: GitHub
- IDE: Visual Studio Code
- Frameworks: Flask and built-in utilities such as Jinja template engine to build user interfaces.
- Database: MySQL
- Password Hashing: Bcrypt
- Encrypt and decrypt data: Fernet
- Hashlib.py (for hashing)
- AWS[1]

[1] Check: https://mosquitto.org/ (Tutor – Cathryn Peoples's advice)

However, due to time constraint, we are still working on the encryption/decryption section of the documents that are downloaded by users. AWS (for the secure hosting of application) has not been looked into yet.

2.4. How to run and execute the code from the server

The server can be started from the command prompt (cmd):

- Copy – Paste the path of the folder where the codes are located in the cmd

- to install dependencies
    - pip install -r requirements.txt
- to populate testing accounts
    - python create_user.py

On MAC:

- export FLASK_ENV=development
- export FLASK_RUN_PORT=8080
- export FLASK_RUN_HOST=0.0.0.0

On Windows:

- set FLASK_ENV=development
- set FLASK_RUN_PORT=8080
- set FLASK_RUN_HOST=0.0.0.0

Running the app:
flask run

To test the application, it is better to run the codes in the development environment:

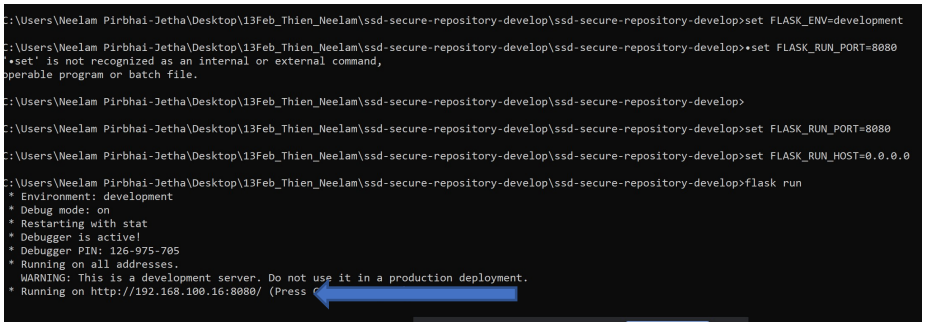


Figure 1: Development server link

The production environment is for deployment:

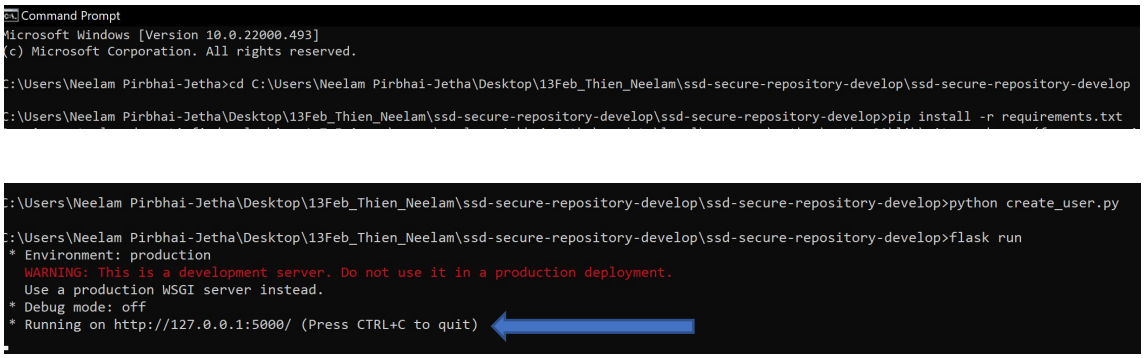


Figure 2: Production server link

To have access to the application, the address (either on development or production environment) must be copied and pasted on the browser.

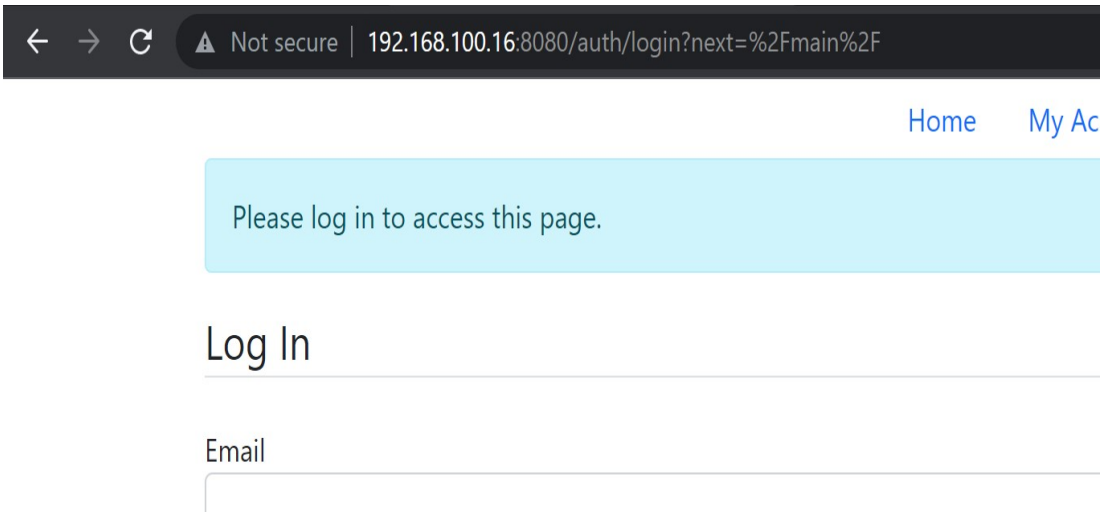


Figure 3: Login page of repository

Users can login in, log out and manage their accounts. As we will see below, the admin can track the users' activities.

2.5. The Security Features

Research shows that most security issues arise from software vulnerabilities, especially in its “configuration, design, and implementation”, and the “lack of knowledge about security concerns” (Medeiros et al, 2017)

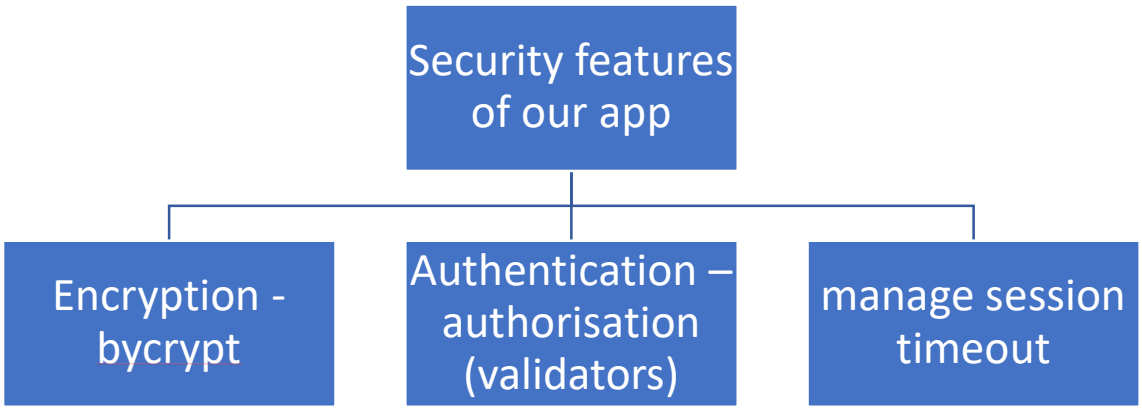


Figure 4: Security Features of the Repository

2.5.1. Encryption-Decryption-Hashing

Authentication reduces attacks from the inside and determines whether the right person is getting access to the authorised information, thus ensuring ‘information security' (preservation of confidentiality, integrity, non-repudiation, reliability) (ISO). Moreover, to prevent ‘Cryptographic Failures’, previously called Sensitive Data Exposure (OWASP, 2021), user’s data must be encrypted and decrypted using proper strategies and tools.

2.5.1.1. Bcrypt

```
"""Flask-Bcrypt is a Flask extension that provides bcrypt hashing utilities:
to protect passwords for instance
Then below add 'Bcrypt=... ' to initialise it in the app; then go to route.py
"""
from flask_bcrypt import Bcrypt
```

create_user.py

```
import email
from app import db, bcrypt
from app.auth.models import User
import sys

def main():
    password = "Qwerty@123"
    hashed_password = bcrypt.generate_password_hash(password)
    user1 = User(
        name = "Thien",
        email = "thien@awesomerepo.com",
        password = hashed_password,
        role = 1,
        status = 2
    )
    user2 = User(
        name = "Neelam",
        email = "neelam@awesomerepo.com",
        password = hashed_password,
        role = 1,
        status = 2
    )
    user3 = User(
        name = "Created Internal Staff",
        email = "internal_created@awesomerepo.com",
        password = hashed_password,
        role = 2,
        status = 1
    )
    db.session.add(user1)
    db.session.add(user2)
```

Figure 5: Bcrypt and hashed passwords

Bcrypt was used to hash the password to make sure it is not stored in plain text; in case the data is exposed, the actual password is still being protected.

2.5.1.2. Flask-hashing

Flask-Hashing, a Flask extension that uses ‘hashlib’, is used to hash – that is to protect and encrypt – data.

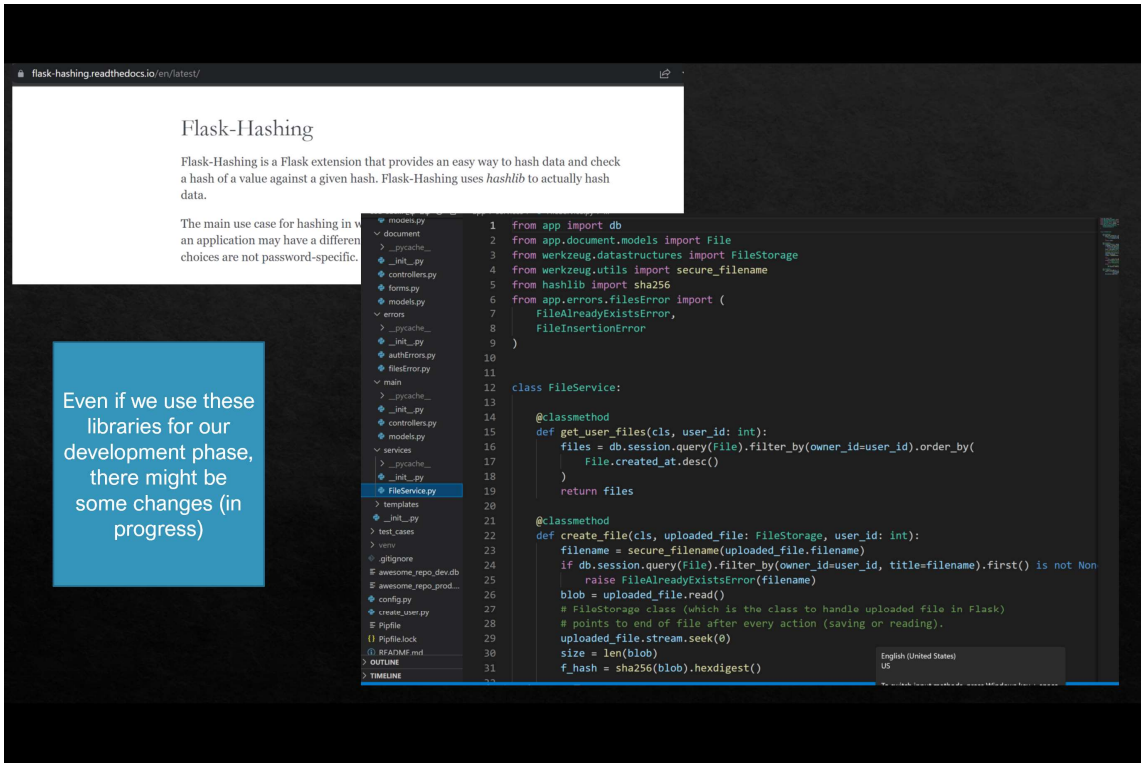


Figure 6: Flask-Hashing

2.5.1.3. Encrypting documents

Downloading documents that have not been encrypted can be create security issues:

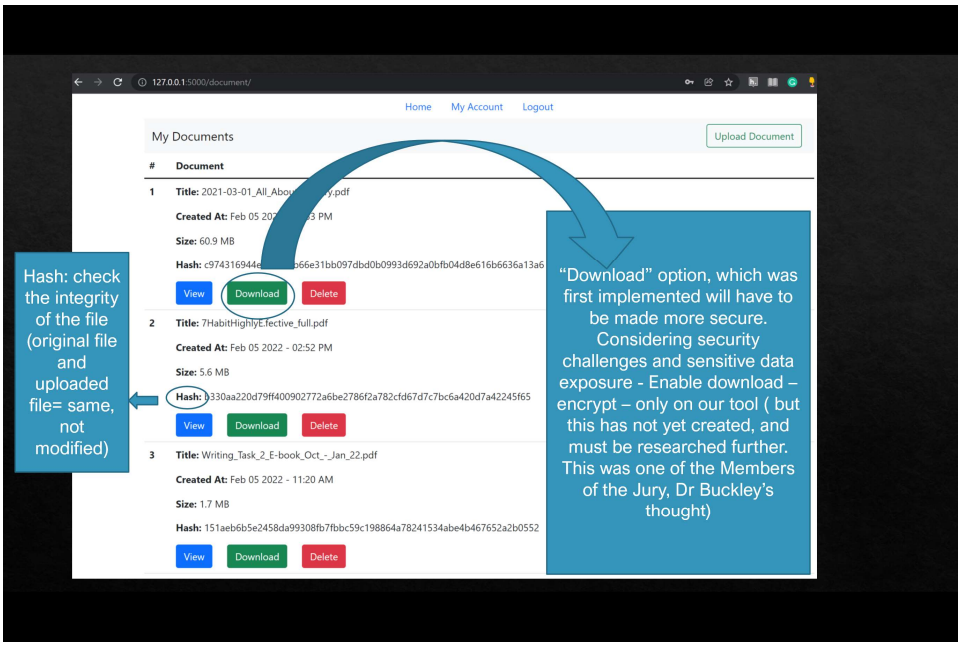


Figure 7: Hashing and secure Download Button

Even if encryption/decryption on the documents being uploaded has not been implemented in our Flask app yet, research is being done on the question. Different encryption algorithms (Peoples, 2021: Unit 8; Tutorialspoint, 2021) are being looked into:

Cypher Algorithms	Drawbacks/Security Vulnerabilities	Advantages
Algorithm of Reverse Cipher	Very weak and easily hacked	
Caeser Cypher	Can be hacked by Brute Force Technique (i.e trying every possible encryption key)	
ROT13 Cypher (Rotate by 13 places)	Easy to hack – by shifting 13 places in reverse	
Transposition Cipher (columnar)		Better than the previous three+re-encrypting the cypher text using the transposition cypher creates better security
Base64 algorithm	Main drawback: stores password in a database	
XOR algorithm		Hard to crack by brute force method
Affine cipher		Includes two functions for encryption and decryption – harder to hack
Monoalphabetic Cipher	Letters change, but frequency doesn't. Therefore, can be cracked by using frequency table	Hard to crack by brute force method
Simple Substitution Cipher		Lots of combinations of letters
Vignere Cipher/polyalphabetic Cipher (One Time Pad Cipher (a type of Vignere Cipher)	Can be cracked if hacker knows the Kasiski method or the Friedman test	Very difficult to hack – considered a secure encryption method Onetime Pad Cypher: almost unbreakable
Data Encryption Standard (Symmetric Cryptography)		Simple and faster; both parties exchange the key in a secure way
RSA Algorithm (Asymmetric Cryptography): has 2 keys – public and private		Most secure way of encryption: Brute force attack won't work – too many keys to try; only numeric so dictionary attack not possible; frequency analysis not possible; no specific mathematical tricks

Table 2: An overview of the different cryptography algorithms

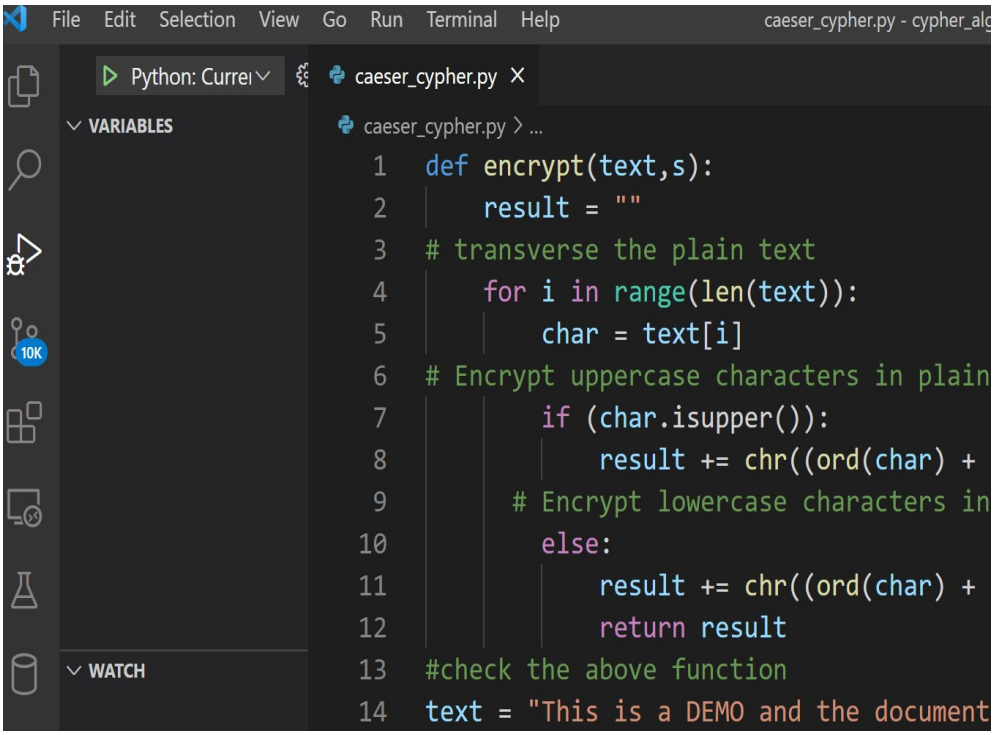


Figure 8: Caesar Cypher Algorithm – trial 1

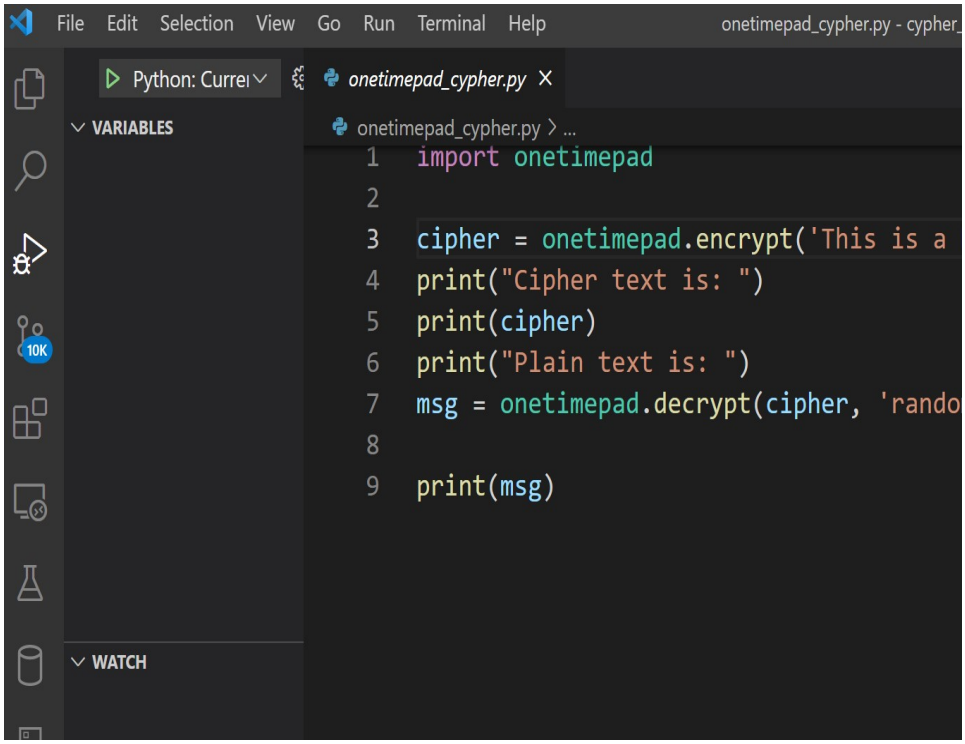


Figure 9: One-time pad algorithm – trial 1

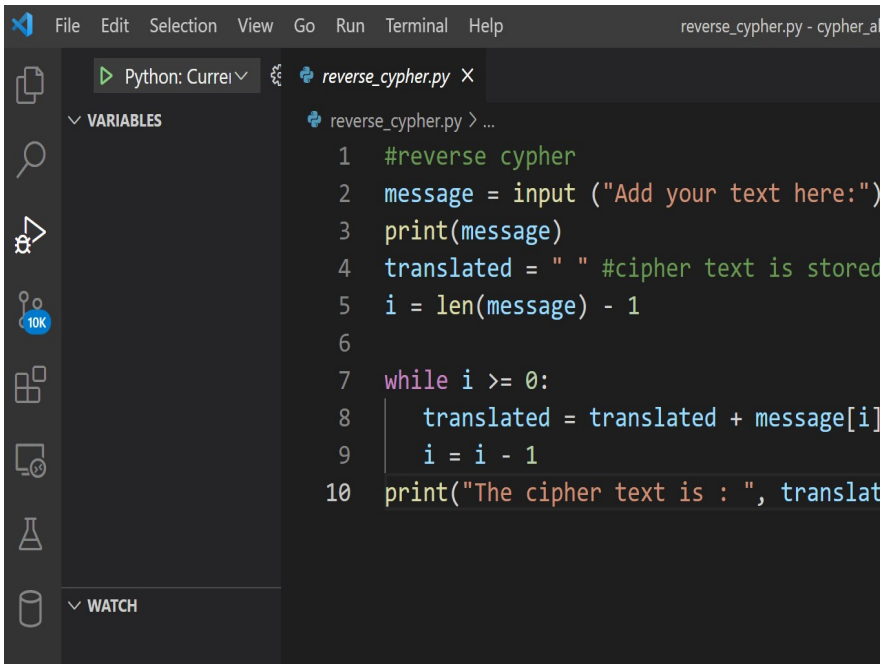


Figure 10: Reverse Cypher Algorithm – trial 1

It seems the best algorithm is the RSA algorithm. Even if it is a bit complicated to implement, it might be better to protect from SQL Injection and Broken Authentication. This will be looked into in the near future.

2.5.2. Managing session time-out

Security issues can occur if session time-out is not “set to the minimal value possible depending on the context of the application” (OWASP, 2022). The session.permanent is set to True after the user successfully logs in.

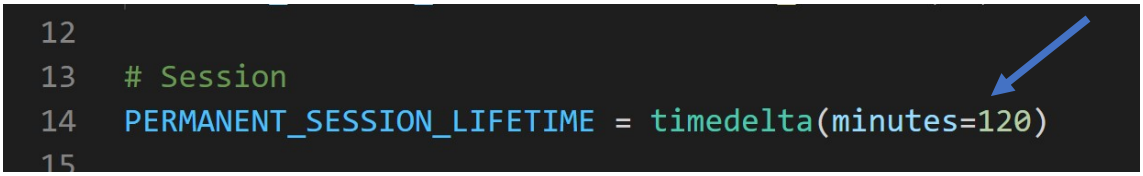


Figure 11: Session Timeout updated

2.5.3. Environmental configurations

The app has three different environmental configurations. The development/testing environment will not go public to the world outside. At the business level, a VPN connection will be required to access the resources from these environments. We will also be using HTTPS (which is HTTP with encryption) (Arampatzis, 2020). A Virtual Private Network (more secure encryption than HTTPS) will give more privacy. Modifications in the development configuration will not affect the others. New features or issue fix can be done in the development configuration, and then deployed to the testing config environment to verify if they pass the testing, then go public for the user (production config-deployment).

Figure 12: Environmental Configurations

2.5.4. Users and Login Manager

The proposed design will follow the different recommendations and best practices in use (OWASP, STRIDE, ISO) to certify that the system meets the most updated security standards. According to OWASP (2021), Broken Access Control has been the most common security risk, which possibly leads to sensitive data disclosure, unauthorised data modification or destruction.

```
"""
pip install flask-login in cmd
then import LoginManager
"Flask-Login provides user session management for Flask.
It handles the common tasks of logging in, logging out,
and remembering your users' sessions over extended periods of time."[Flask Login Docume
"""
from flask_login import LoginManager
```

```
# Import the database object from the main app module
from app import db, bcrypt, login_manager

from flask_login import login_user, current_user, logout_user, login_required, LoginManager

from app.auth.forms import LoginForm, ChangePasswordForm
from app.auth.models import User

auth = Blueprint('auth', __name__, url_prefix='/auth')
app_config = current_app.config

# A user_loader callback, used to reload the user object
# from the user ID stored in the session
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

@auth.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        if current_user.isAdmin:
            return redirect(url_for('admin.home'))
        else:
            return redirect(url_for('main.home'))

    form = LoginForm(request.form)

    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
```

Figure 13: Login Manager in Flask

We have used Flask-Authorize as this library in Python can secure an app, by giving different roles to users. (Deny by default, Enforce Least Privileges).

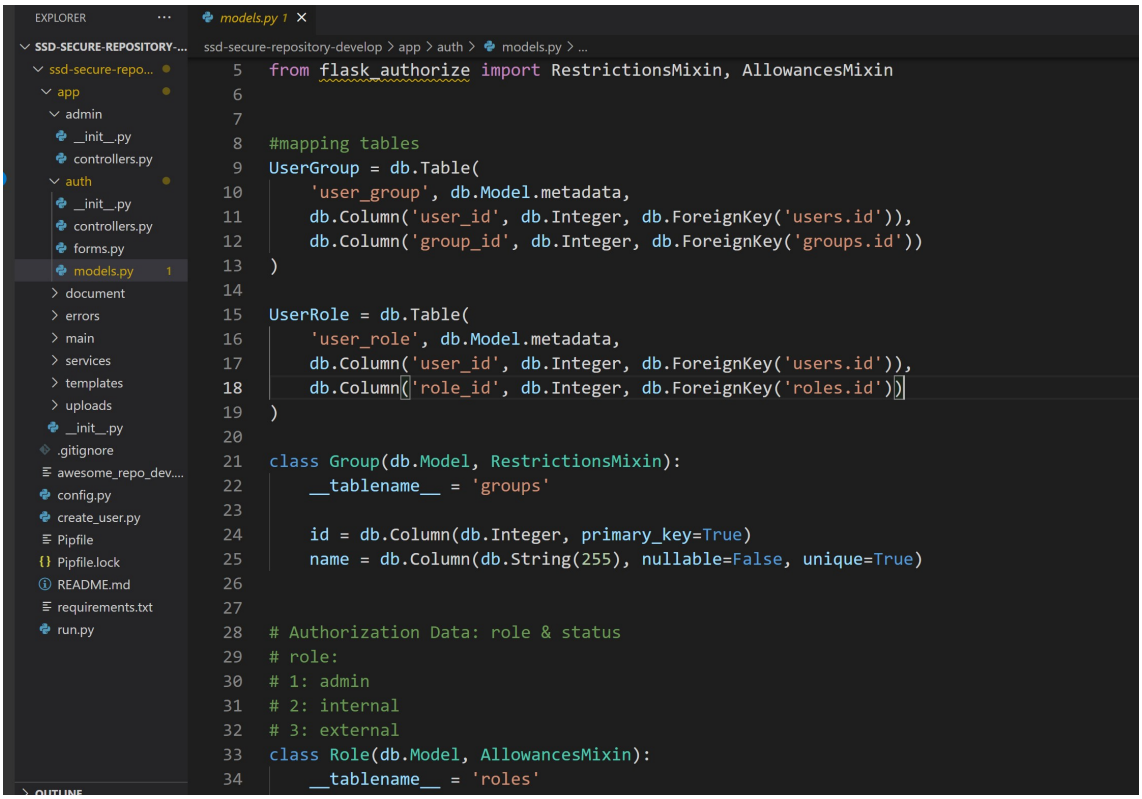


Figure 14: Flask-Authorize

2.5.5. Regex

Figure 15: Regular Expressions

On Flask, email validators are used instead of typing the Regex codes:

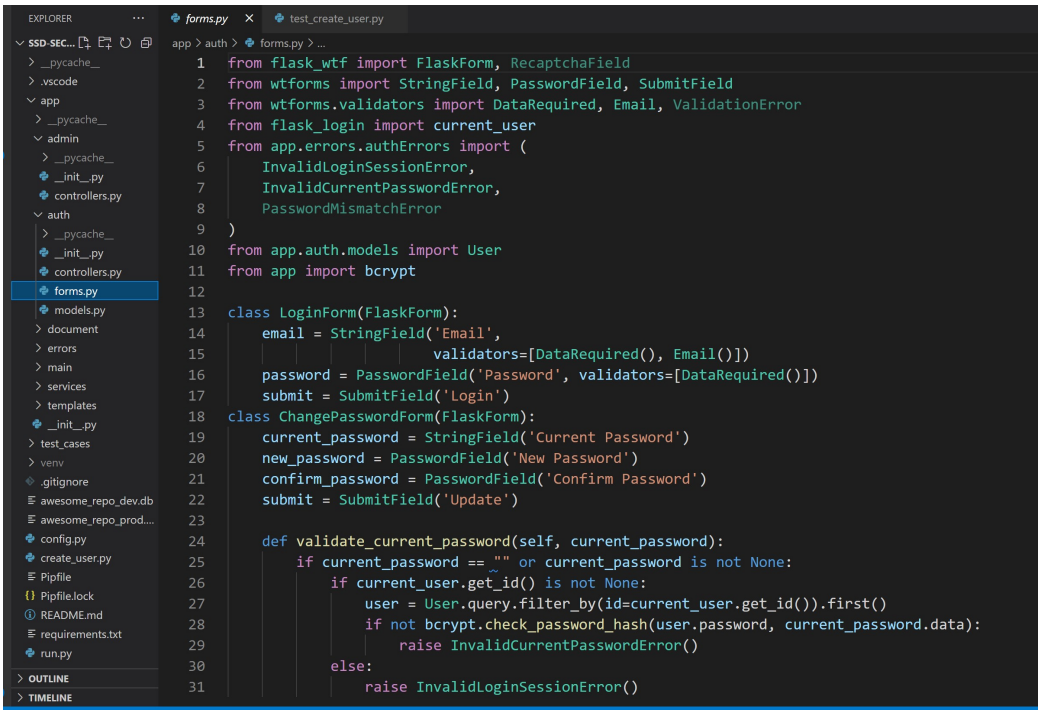


Figure 16: Email Validators instead of Regex

2.5.6. Generating Secret Keys

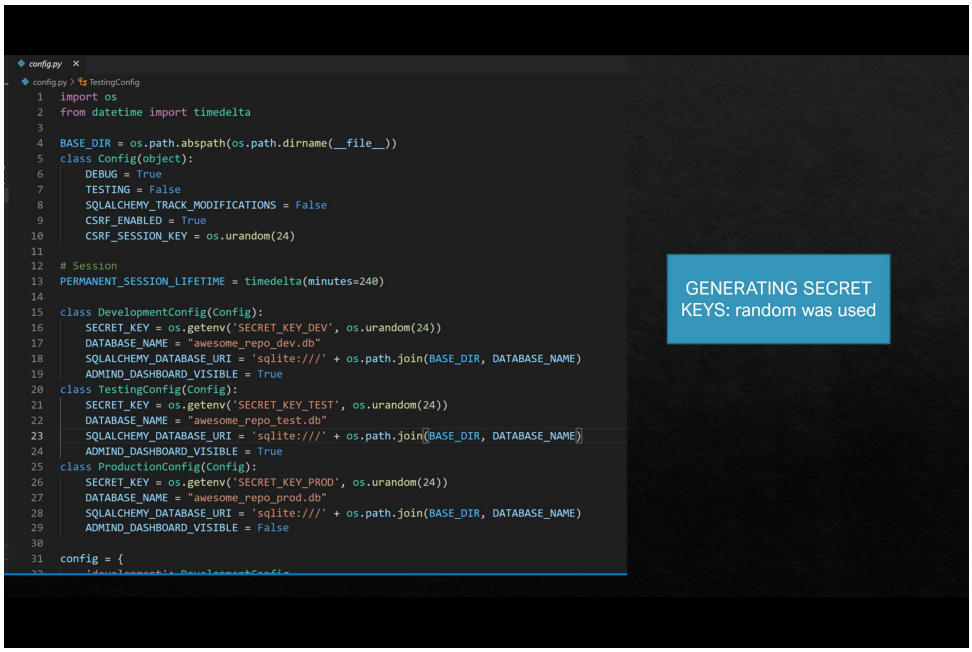


Figure 17: Secrets Keys with Random

A one-time password (OTP) will be added to the Flask app later. The codes were at first written with Random. However, random generator is not secure:

Figure 18: OTP with Random (not secure)

Secrets has been used instead of Random:

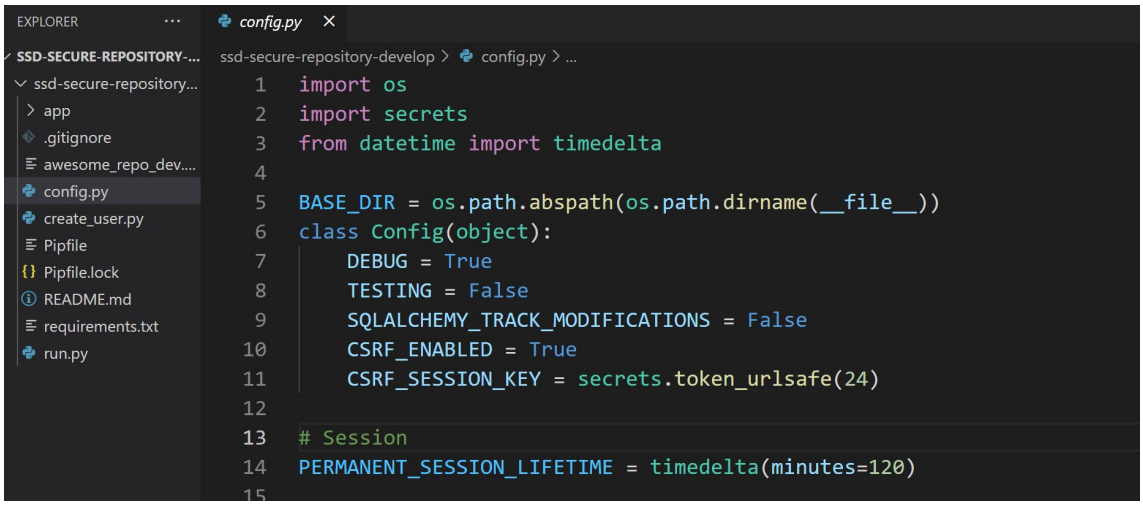


Figure 19: Secrets used instead of Random

2.5.7. Raising errors

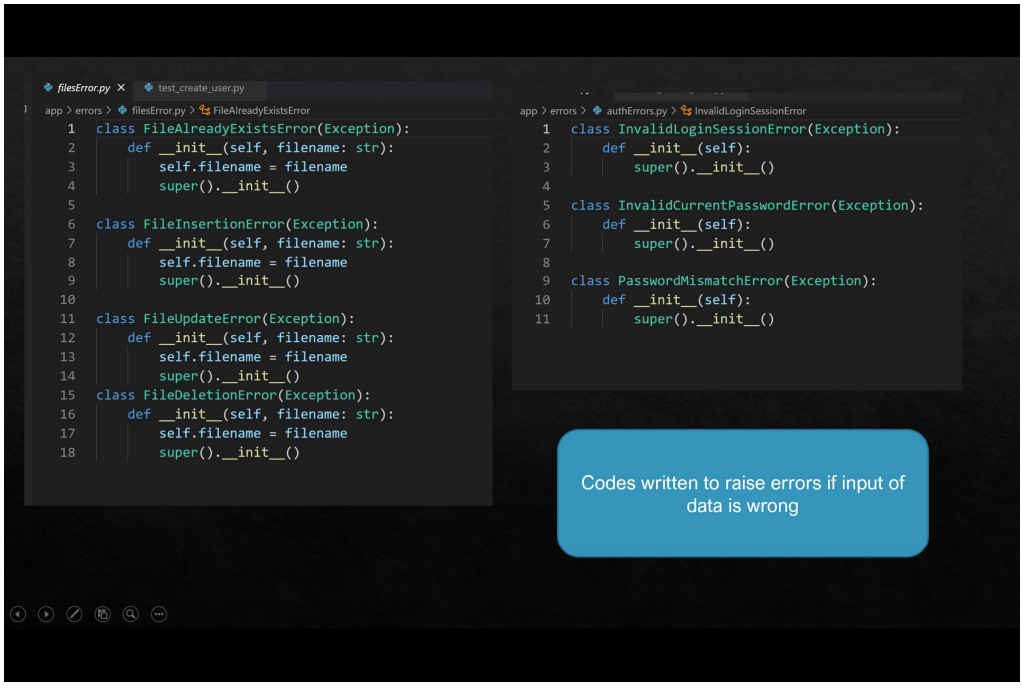


Figure 20: Wrong data input will raise errors

2.5.8. Activity Logs/Admin Dashboard

Apart from the One-time password that has not been implemented, our system has tried to follow the design codes to:

- implement strict user authentication and authorisation policies
- fulfill a user role based and hierarchical authentication model
- operate over secure web-connections and enable monitoring of user activities via systems audit logs
- delete a user upon request (GDPR's right to be forgotten)

As shown below, logging in as Admin gives access to the Activity Logs, which keep tracks of all the users' activities (view document, fail to upload or delete file)

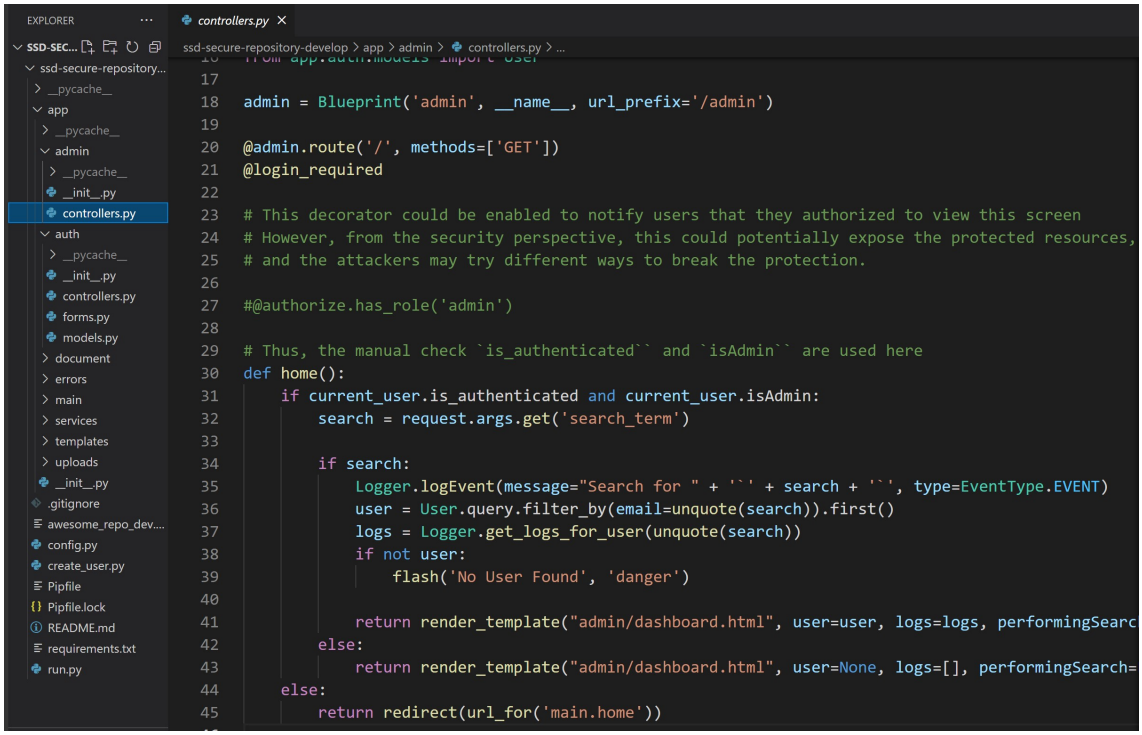
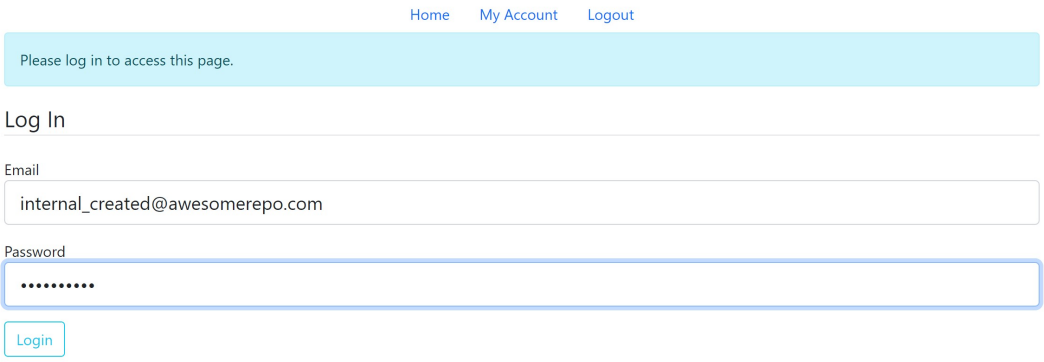


Figure 21: codes for Admin_dashboard

To understand how the dashboard/activity logs functions, log in as ‘internal user’ with the following credentials:

internal_created@awesomerepo.com

Qwerty@123



Then upload any document:

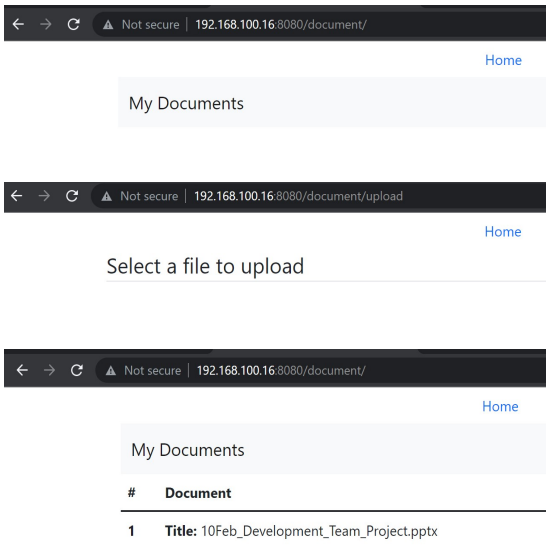
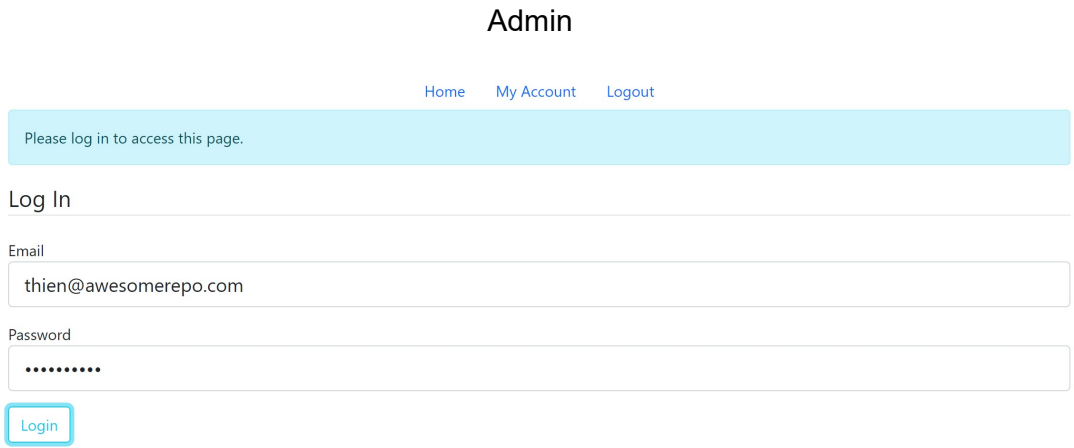


Figure 22: users' signing in and uploading a document

The security vulnerability that will have to be corrected is the uploading of documents from the desktop without any control. Discussion is ongoing on whether to add an OTP or to find another solution.

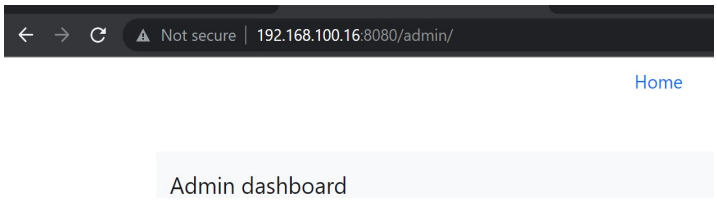
To have access to the dashboard, use the following credentials to login as admin:

```
thien@awesomerepo.com
Qwerty@123
```



Then search for a user by typing his/her email:

(internal_created@awesomerepo.com)



The dashboard gives details about day/date/time user logged in and the IP Address:

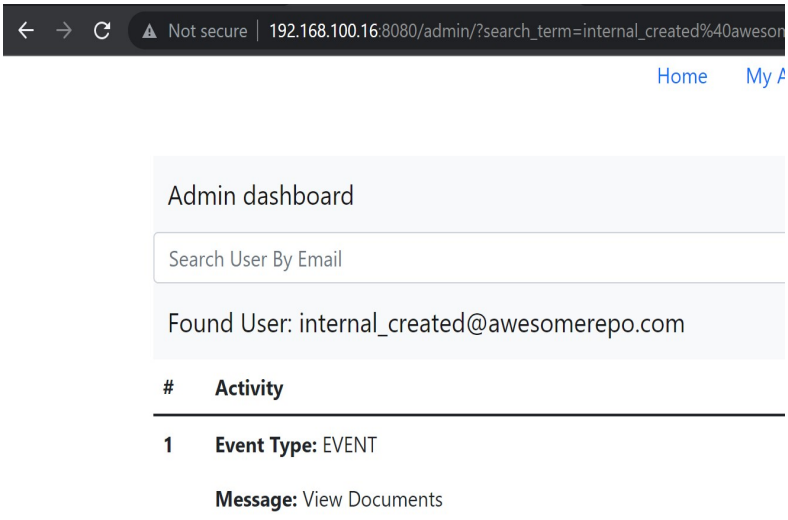


Figure 23: The user's activities on the admin dashboard (Activity log)

2.5.9. GDPR 'right to be forgotten'

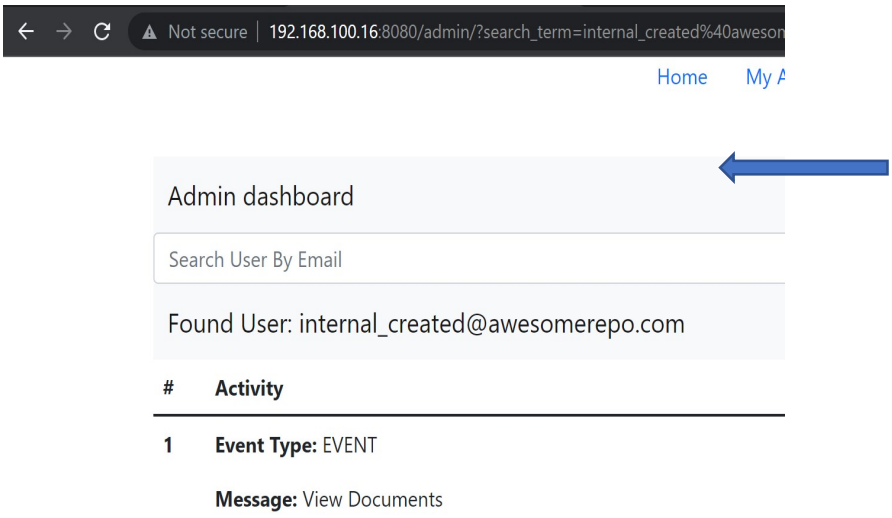


Figure 24: GDPR right to be forgotten

Users, along with all their activities, can be deleted, thus complying to the GDPR regulations. This dashboard, still in its embryonic stage, will later be extended to generate/edit user or assign permissions among others.

2.6. TESTING

2.6.1. Test Data

The table below, created in more detailed during the design period, explains the different vulnerabilities that could be encountered during the implementation. It can enable us to check if some software flaws and bugs have been taken into consideration to mitigate any damage:

Threat	Comments (libraries used/action taken)
Broken access control	● Flask-authorize
Sensitive data exposure	● Hashlib ● Bycrypt
Denial of Service (Erickson)	● Don't have the configs yet ● API rate limit – too many requests will raise an error
Injection	● email-validators
Buffer overflow	● Investigate Flask-authorize, Bcrypt, etc. and check if there is no buffer-overflow in the existing library

Table 3: Python libraries against vulnerabilities

2.6.2. Quality Control

We have tried to apply some of the tools such as pylint, flake8, mccabe, cyclomatic complexity (Peoples, 2021).

2.6.2.1. Manual Error Handlers

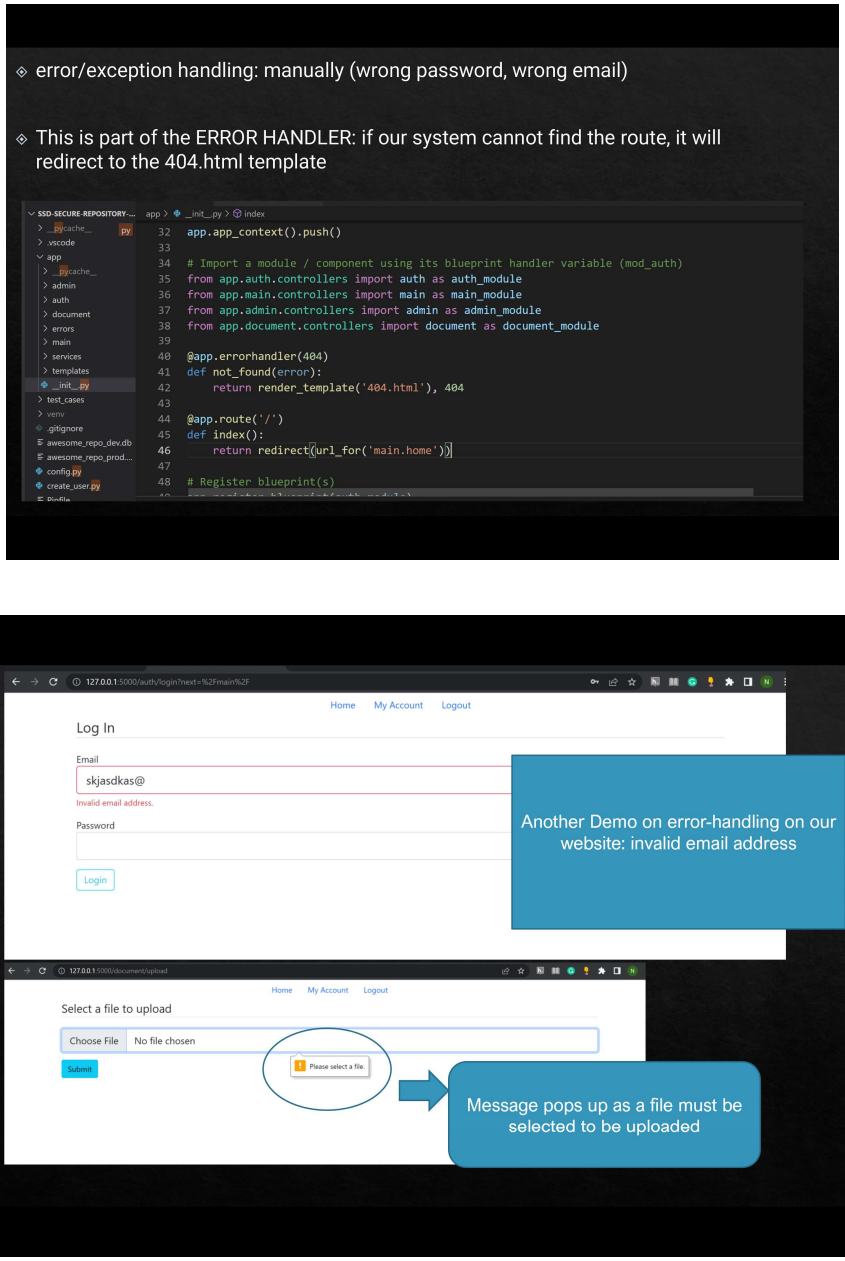
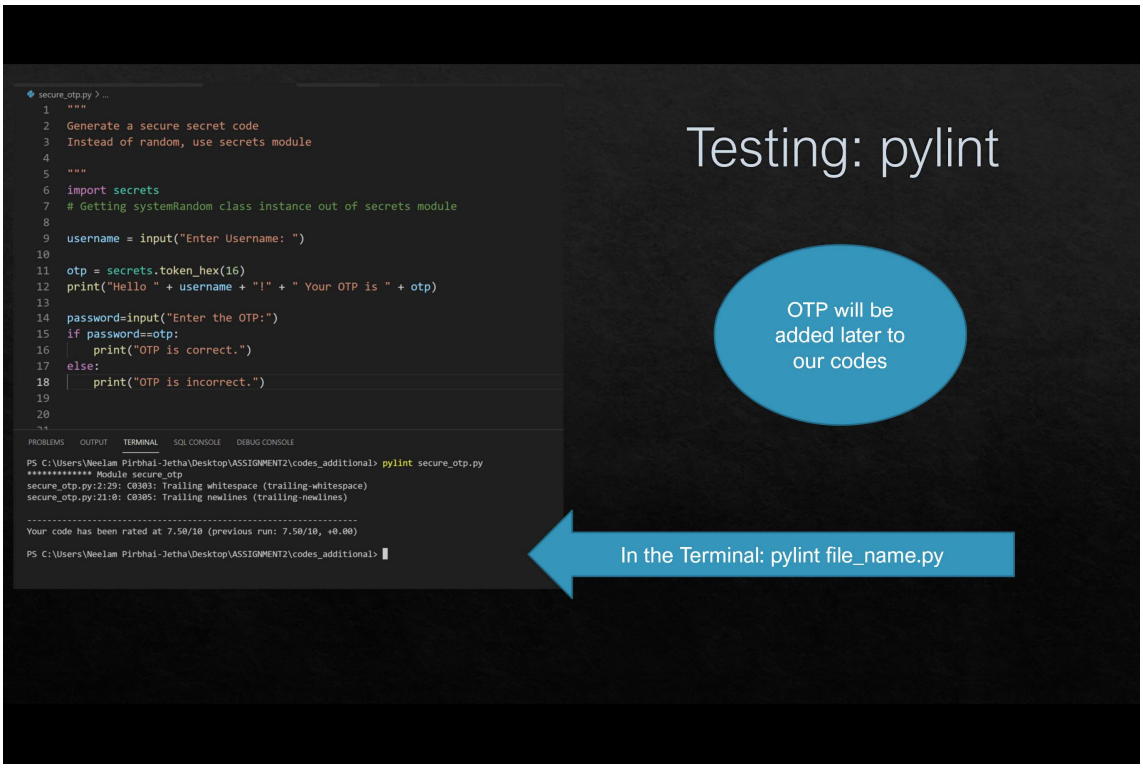
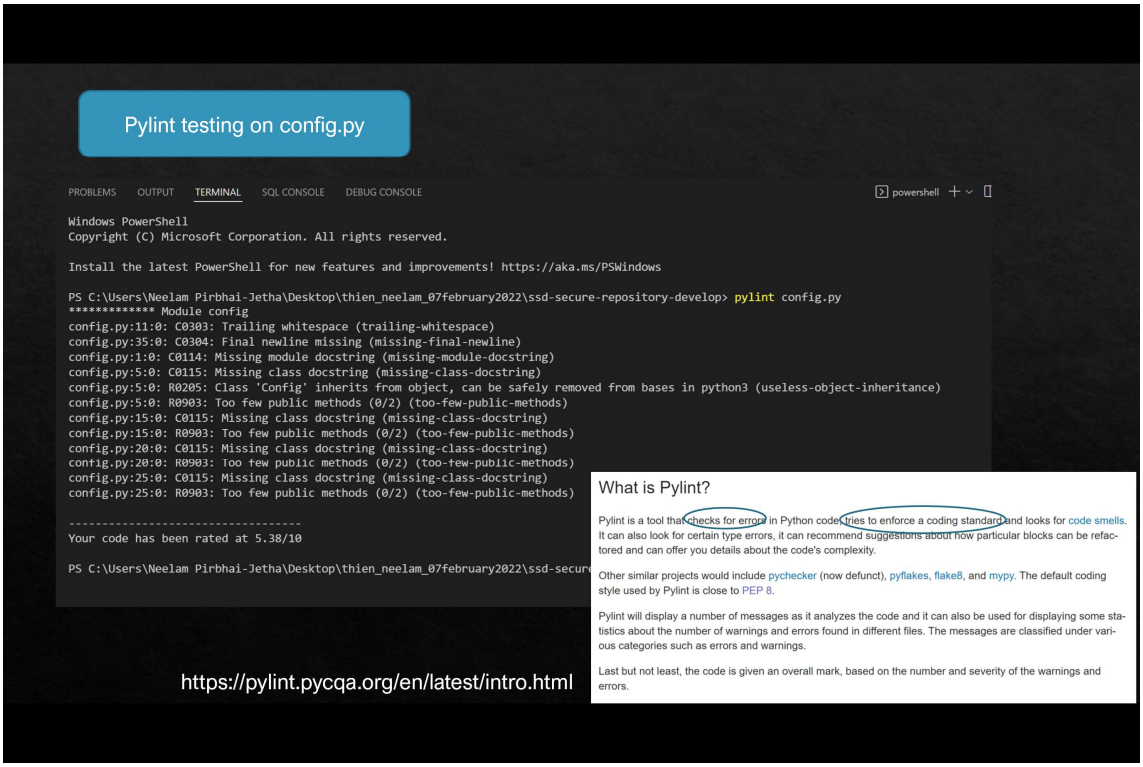


Figure 25: Manual error handling

2.6.2.2. Pylint


secure_otp.py > ...
"""
Generate a secure secret code
Instead of random, use secrets module
"""
import secrets
# Getting systemRandom class instance out of secrets module
username = input("Enter Username: ")
otp = secrets.token_hex(16)
print("Hello " + username + "!" + " Your OTP is " + otp)
password=input("Enter the OTP:")
if password==otp:
print("OTP is correct.")
else:
print("OTP is incorrect.")
PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional> pylint secure_otp.py
************ Module secure_otp
secure_otp.py:2:29: C0303: Trailing whitespace (trailing-whitespace)
secure_otp.py:21:0: C0305: Trailing newlines (trailing-newlines)
Your code has been rated at 7.50/10 (previous run: 7.50/10, +0.00)
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional>
Testing: pylint
OTP will be added later to our codes
In the Terminal: pylint file_name.py


Pylint testing on config.py
PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> pylint config.py
************ Module config
config.py:11:0: C0303: Trailing whitespace (trailing-whitespace)
config.py:35:0: C0304: Final newline missing (missing-final-newline)
config.py:1:0: C0114: Missing module docstring (missing-module-docstring)
config.py:5:0: C0115: Missing class docstring (missing-class-docstring)
config.py:5:0: R0205: Class 'Config' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
config.py:5:0: R0903: Too few public methods (0/2) (too-few-public-methods)
config.py:15:0: C0115: Missing class docstring (missing-class-docstring)
config.py:15:0: R0903: Too few public methods (0/2) (too-few-public-methods)
config.py:20:0: C0115: Missing class docstring (missing-class-docstring)
config.py:20:0: R0903: Too few public methods (0/2) (too-few-public-methods)
config.py:25:0: C0115: Missing class docstring (missing-class-docstring)
config.py:25:0: R0903: Too few public methods (0/2) (too-few-public-methods)
Your code has been rated at 5.38/10
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure
What is Pylint?
Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for code smells. It can also look for certain type errors, it can recommend suggestions about how particular blocks can be refactored and can offer you details about the code's complexity.
Other similar projects would include pychecker (now defunct), pyflakes, flake8, and mypy. The default coding style used by Pylint is close to PEP 8.
Pylint will display a number of messages as it analyzes the code and it can also be used for displaying some statistics about the number of warnings and errors found in different files. The messages are classified under various categories such as errors and warnings.
Last but not least, the code is given an overall mark, based on the number and severity of the warnings and errors.
https://pylint.pycqa.org/en/latest/intro.html

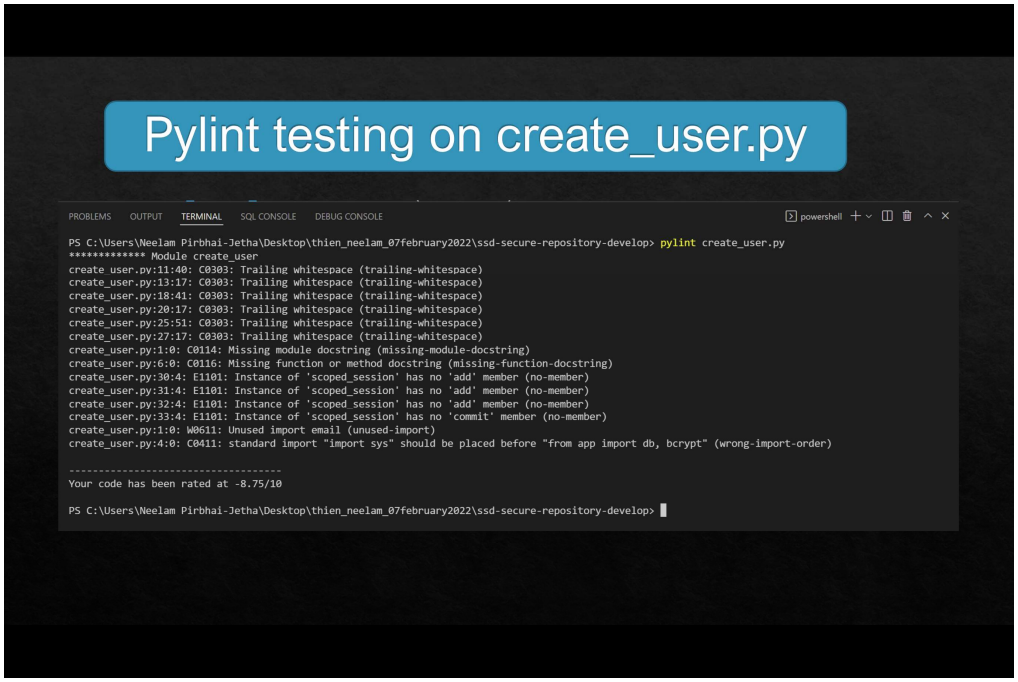


Figure 26: Pylint

2.6.2.3. Flake8

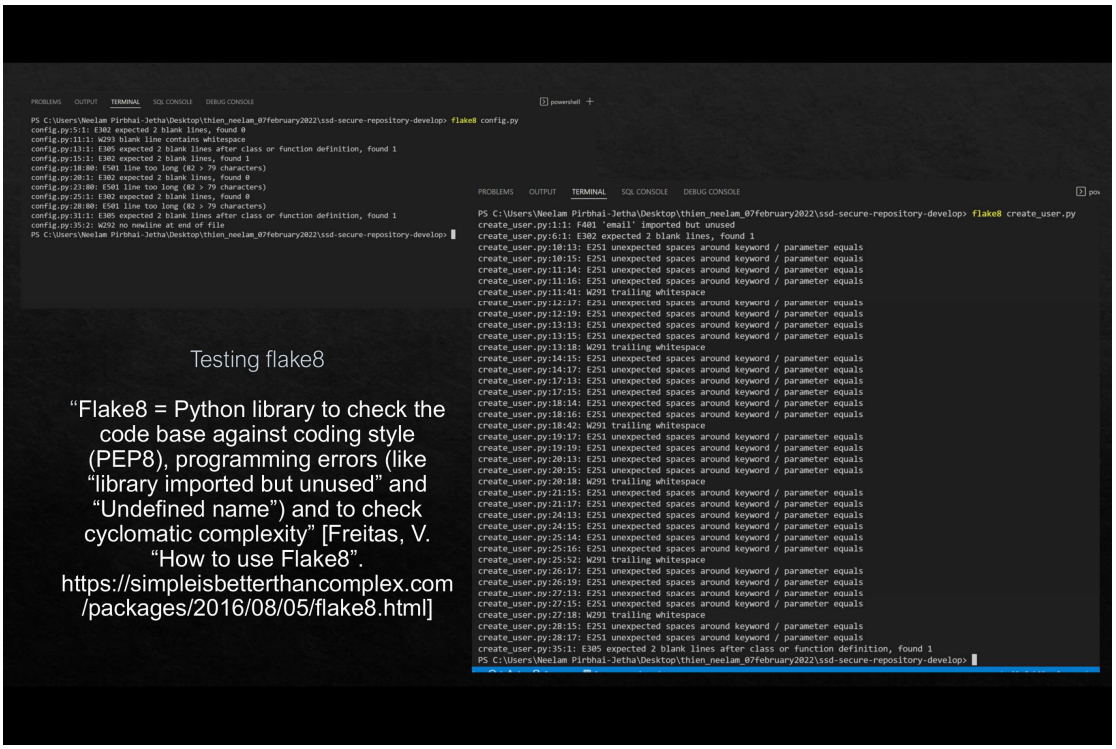


Figure 27: Flake8

2.6.2.4. Mccabe

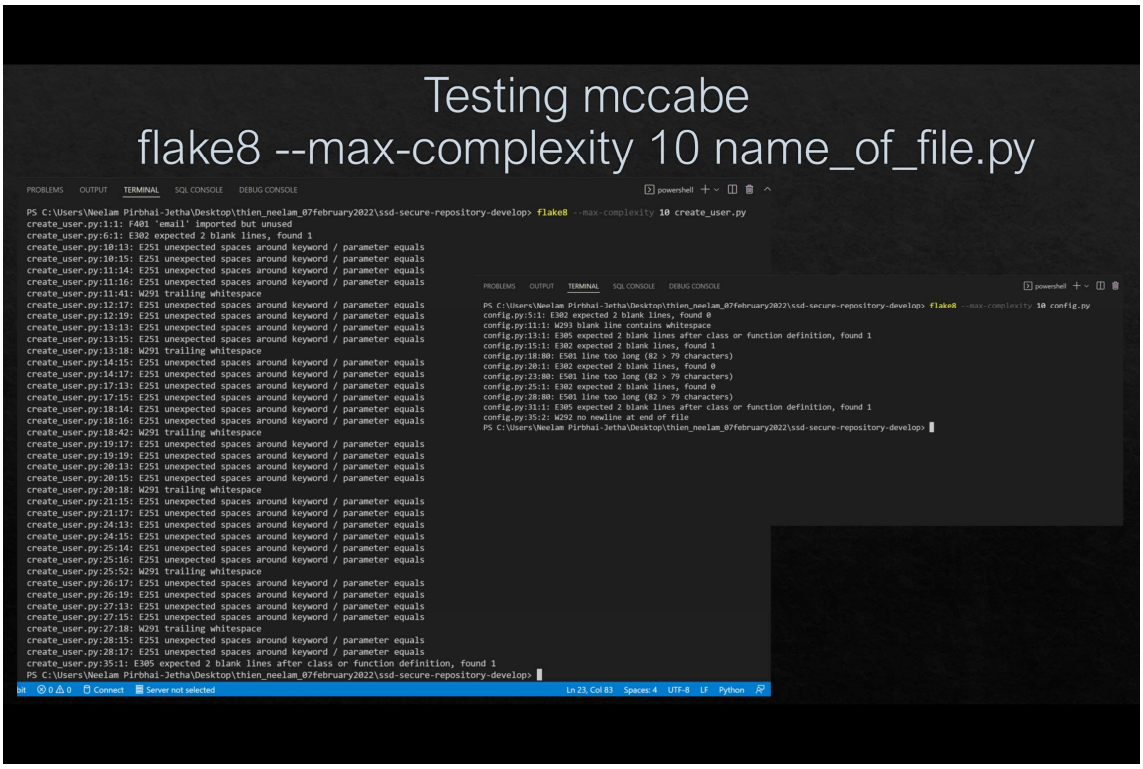


Figure 28: Mccabe

2.6.2.5. Cyclomatic complexity

Radon is a Python tool that computes various metrics from the source code such as:

- **“**McCabe’s complexity, i.e. cyclomatic complexity
- Raw metrics (these include SLOC, comment lines, blank lines, &c.)
- Halstead metrics (all of them)

- Maintainability Index (the one used in Visual Studio)" (Python Software Foundation, 2022)

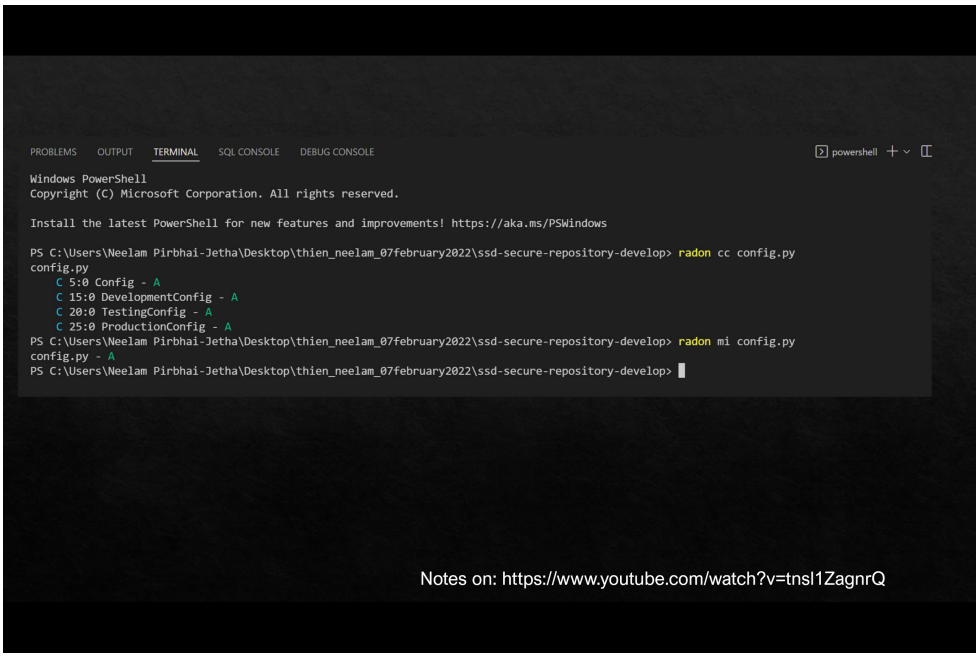


Figure 29: Cyclomatic Complexity Testing

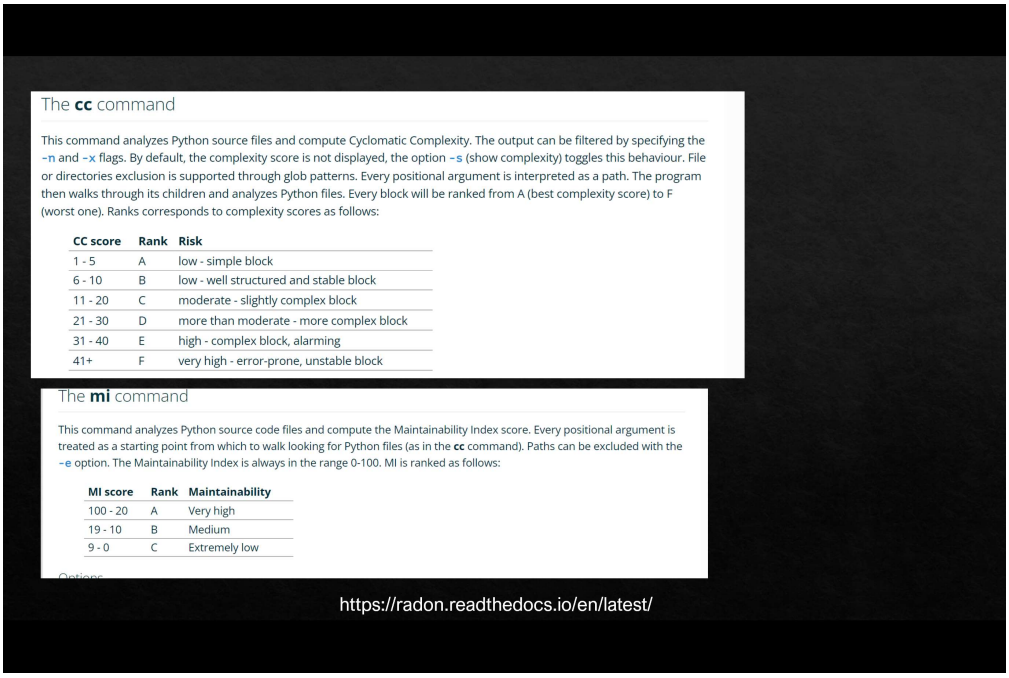


Figure 30: Cyclomatic Complexity Results Explanation

2.6.2.6. Bandit

The linter, Bandit, was installed on VSCode, and all files are being checked automatically.

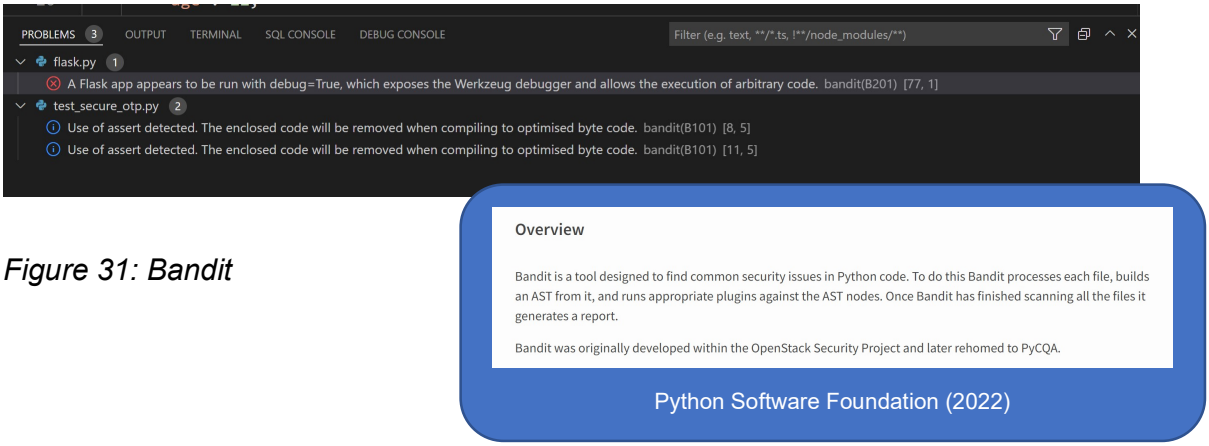
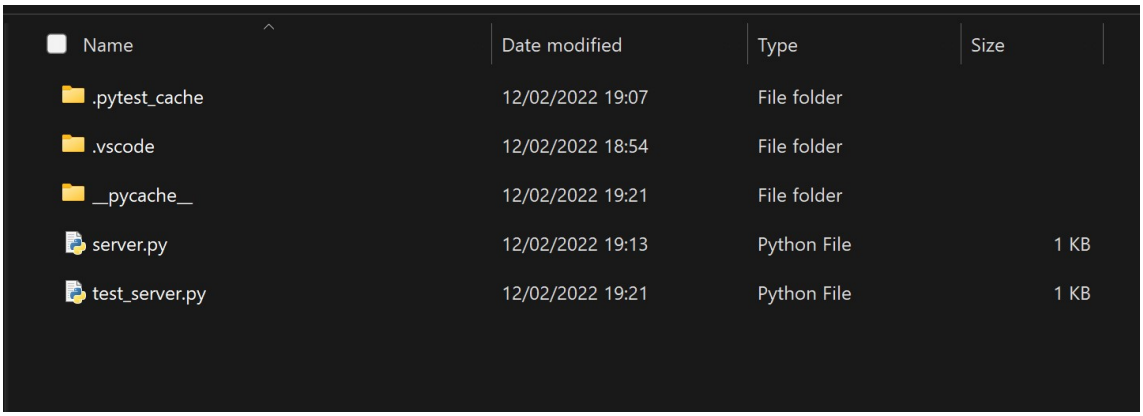


Figure 31: Bandit

2.6.2.7. Other tests

PyTest, a testing framework, mainly used for writing tests for APIs is being researched. Pytest, compared to unittest, is easier to work with. Testing has not been done on the actual app yet. A simple flask app was created to learn about pytest:



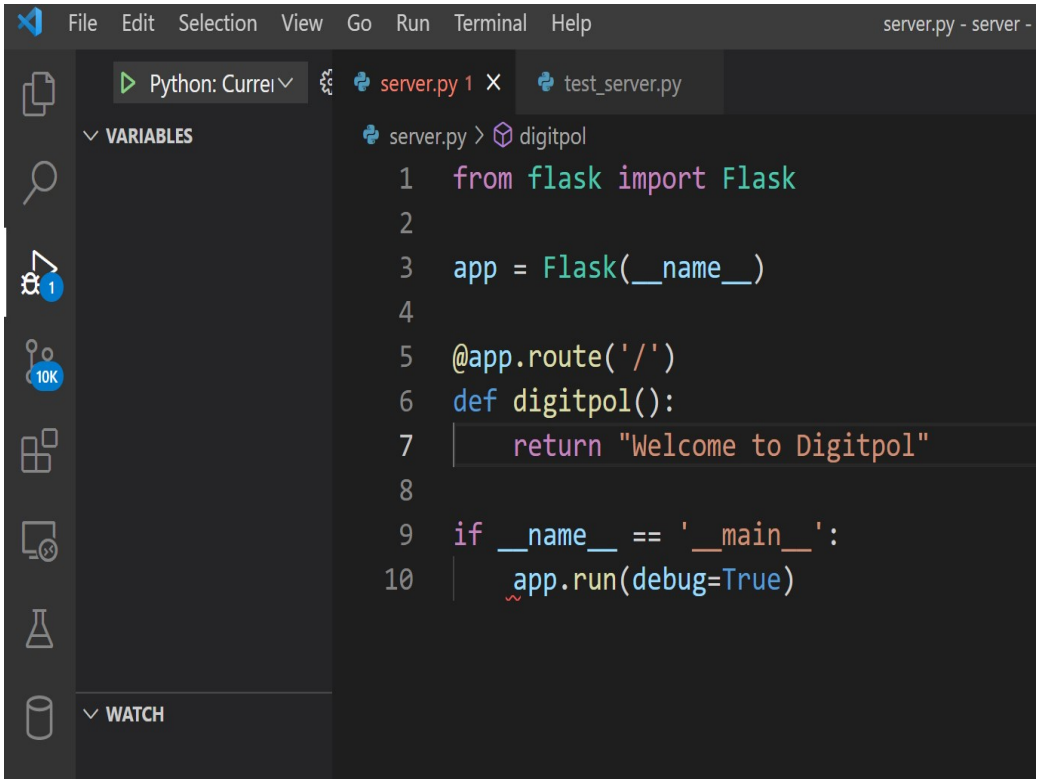


Figure 32: Simple flask app

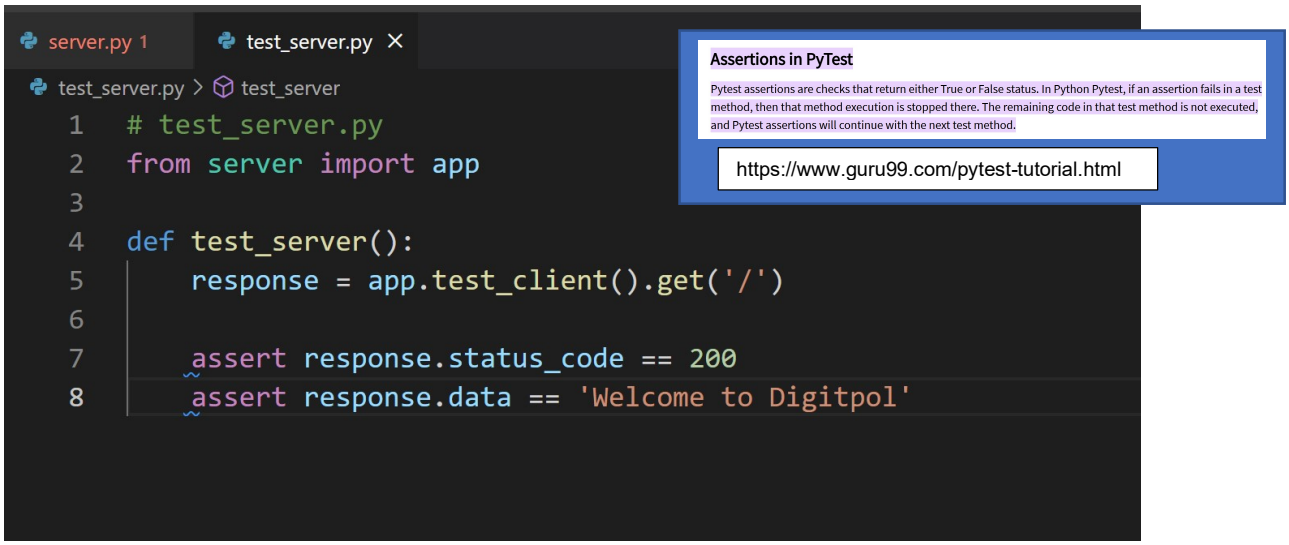


Figure 33: pytest codes

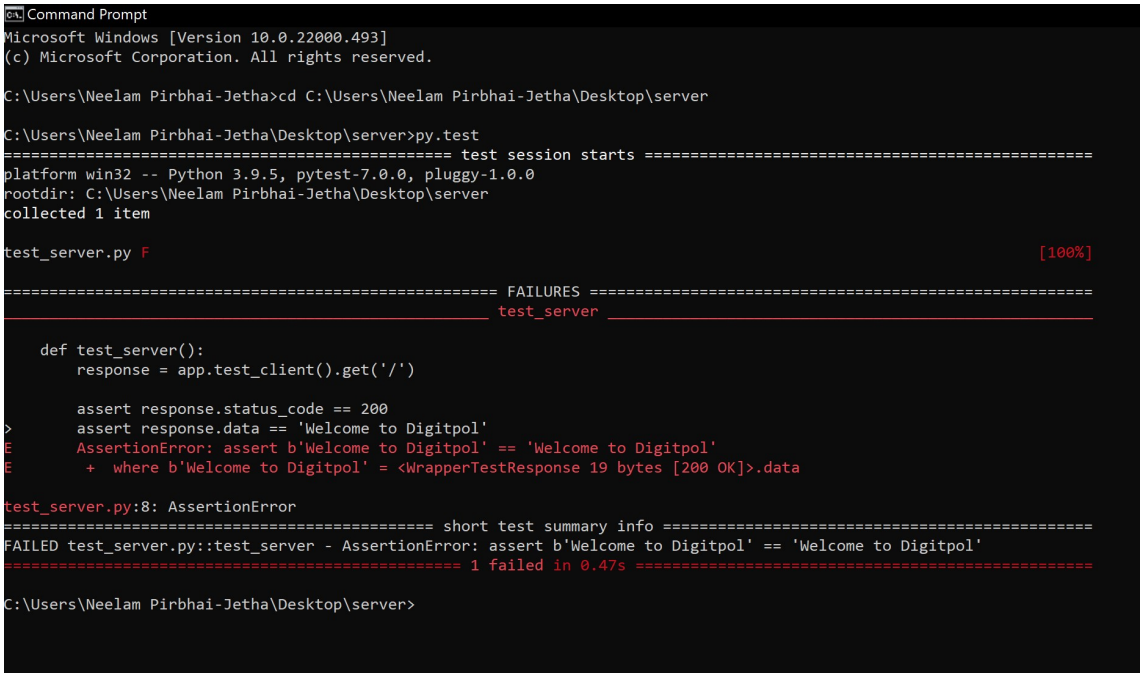


Figure 34: pytest results (in the command prompt)

Equivalence Partitioning Testing or Equivalence Class Partitioning, as a black box testing is used, to test whether the right number of digits have been entered for the OTP, for instance. However, we have yet to research how to use equivalence Testing.

2.7. Summary and Conclusion

In 2018, the General Data Protection Regulation (GDPR) has brought more “control and higher standards” concerning the data of EU residents anywhere in the world. Companies not abiding by the regulations face high financial penalties (Brown, 2018).

Security issues around the designing and developing of an application, therefore, needs to be thought properly. As suggested by many researchers, codes must be

tested early in the software development life cycle (SDLC), thus making fixing any security issues easier and cheaper (Amerding, 2021).

In the development of our repository, we have tried to abide by the GDPR's regulations on data protection: authentication and authorisation will allow only registered users to get access to the repository. An admin dashboard has been added to audit the logs and track the user's activities. This can help detect unusual activities on the repository.

The "rights to be forgotten" or the right to erasure of users (EU-GDPR) question has been looked into. The current admin dashboard has a feature to filter a specific user by email, with all the associated activity logs. After that, an admin can perform a user deletion upon request. All the data associated with the users will be deleted too, including their documents and logs.

While the app was being developed, it became complicated to apply OTP (since mobile/email services had to be used such as twilio). If the development team decides to integrate the one-time password, it would be an optional, stand-alone service that can be later deployed to the application. More security protocols, apart from SQLAlchemy (used in the app), will be researched to avoid SQL injection.

Moreover, further research has to be done on cryptography and how to protect uploaded/downloaded documents. Considering the encryption and decryption of the documents, an external tool may be required to decrypt the file content (such as VitalSource). For the scope of this version, we ensure the path coming from the client is not maliciously crafted to point outside the specified directory (Pallets, 2010).

To sum up, we can say that steps are being taken to secure the repository.

Number of words
(including titles, subtitles, captions, descriptions and discussions)**: 2116**

2.8. References

2.8.1. Reference list for the codes:

Codemy.com (March 12, 2021) “How To Use MySQL Database With Flask - Flask Fridays #9”. Available from: https://www.youtube.com/watch?v=hQl2wyJvK5k

Flask (n.d) “Flask-Bcrypt”. Available from: https://flask-bcrypt.readthedocs.io/en/latest/

Haider, R. (2021) WEB API DEVELOPMENT WITH PYTHON A Beginner’s Guide using Flask and FastAPI.

Tutorialspoint (2015) Flask Web Application Framework. Tutorials Point Pvt. Ltd.

PrettyPrinted (October 12, 2019) “Intro to Flask-Security”. Available from: https://www.youtube.com/watch?v=LsHf3JSDBVc

PythonHow (n.d) “How making a website with Python works”. Available from: https://pythonhow.com/python-tutorial/flask/How-making-a-website-with-Python-works/

Schafer, C. (May 4, 2018) Python Flask Tutorial: Full-Featured Web App Parts 1-12. Available from: https://www.youtube.com/watch?v=MwZwr5Tvyxo&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH + check: https://github.com/CoreyMSchafer/code_snippets/tree/master/Python/Flask_Blog

Sudharakan, S. (March 13, 2019) “How to Use WTForms and Flask-WTF to Create Forms - Chat App Part3”. Available from: https://www.youtube.com/watch?v=EpJRJsmqnn0

Tech with Tim (July 27, 2021) “Python Blog Tutorial #2 - Flask User Authentication and Security”. Available from: https://www.youtube.com/watch?v=W4GItcW7W-U

Traversy Media (April 26, 2017) “Python Flask from scratch”. Available from: https://www.youtube.com/watch?v=zRwy8gtgJ1A

2.8.2. Reference list on security among others

Amerding T. (21 January, 2021) “How to evaluate the ROI of your software security program”. Available from: https://www.synopsys.com/blogs/software-security/evaluating-roi-software-security/ [Accessed 12 February 2022]

Arampatzis, A. (December 10, 2020) “What Are the Differences Between HTTP and HTTPS?” Available from: https://www.venafi.com/blog/what-are-differences-between-http-https-0 [Accessed 12 February 2022]

Brown, A. (2018) “The 7 elements of GDPR software security compliance”. Synopsys Inc. Available from: https://www.synopsys.com/content/dam/synopsys/sig-assets/checklist/7-elements-gdpr-software-security.pdf [Accessed 12 February 2022]

Chopade, R. & Pachghare V.K. (2019) “Ten years of critical review on database forensics research”, *Digital Investigation*. Elsevier. Available from: https://doi.org/10.1016/j.diin. 2019.04.001 [Accessed 25 November 2021]

DIGITPOL (2021) *Netherlands Digital Forensic Investigation - DIGITPOL*. Available from: https://digitpol.com/netherlands-digital-forensic-investigation/. [Accessed 20 November 2021].

Flask (n.d) “Flask-Authorize”. Available from: https://flask-authorize.readthedocs.io/en/latest/ [Accessed 12 February 2022]

Flask (n.d) “Flask-hashing”. Available from: https://flask-hashing.readthedocs.io/en/latest/

ISO. (N.D) ISO/IEC 27000:2018: “3 Terms and Definitions”. Available from: https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en [Accessed 16 November 2021].

Kennedy, P. (December 4, 2021) “Testing Flask Applications with pytest”. FlaskCon. Available from: https://www.youtube.com/watch?v=OcD52lXq0e8 [Accessed 12 February 2022]

Leach, R.J. (2016) *Introduction to Software Engineering*. 2nd ed. London: CRC Press/Taylor and Francis Group.

Medeiros, N. et al (2017) "Software Metrics as Indicators of Security Vulnerabilities," *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*. Available from: doi: 10.1109/ISSRE.2017.11. [Accessed 16 December 2021]

National Cyber Security Centre (2022) “General Data Protection Regulation (GDPR)” Available from: https://www.ncsc.gov.uk/information/GDPR [Accessed 12 February 2022]

OWASP (2022) “Session Timeout”. Available from: https://owasp.org/www-community/Session_Timeout [Accessed 12 February 2022]

OWASP (Open Web Application Security Project). (2021) *C7: Enforce Access Controls*. Available from: https://owasp.org/www-project-proactive-controls/v3/en/c7-enforce-access-controls. [Accessed 20 November 2021].

OWASP. (2021) *OWASP Top Ten Web Application Security Risks*. Available from: https://owasp.org/www-project-top-ten/ [Accessed 20 November 2021].

Pallets (2010) “Testing Flask Applications”. Available from : https://flask.palletsprojects.com/en/2.0.x/api/#flask.send_from_directory [Accessed 13 February 2022]

Peoples, C. (2021) Lecture Notes. University of Essex Online November 2021.

Pillai, A.B. (2017) Software Architecture with Python, Birmingham, Packt.

Python Software Foundation (2022) “bandit 1.7.2”. Available from: https://pypi.org/project/bandit/ [Accessed 12 February 2022]

Python Software Foundation (2022) “radon 5.1.0”. Available from https://pypi.org/project/radon/ [Accessed 9 February 2022]

Tutorialspoint (2021) “Cryptography with Python Tutorial”. Available from: https://www.tutorialspoint.com/cryptography_with_python/index.htm [Accessed 12 February 2022]