

## **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

1.	Introduction.....	4
1.1.	A brief overview.....	4
1.2.	Planning the development of the application .....	4
2.	README FILE .....	5
2.1.	Title.....	5
2.2.	Description .....	5
2.3.	Development Tools and Libraries:.....	5
2.4.	How to run and execute the code from the server.....	6
2.5.	The Security Features .....	8
2.5.1.	Encryption-Decryption-Hashing .....	8
2.5.1.1.	Bcrypt .....	9
2.5.1.2.	Flask-hashing .....	10
2.5.1.3.	Encrypting documents .....	11
2.5.2.	Managing session time-out .....	14
2.5.3.	Environmental configurations.....	15
2.5.4.	Users and Login Manager.....	16
2.5.5.	Regex .....	18
2.5.6.	Generating Secret Keys.....	20
2.5.7.	Raising errors.....	21
2.5.8.	Activity Logs/Admin Dashboard .....	22
2.5.9.	GDPR ‘right to be forgotten’ .....	25
2.6.	TESTING.....	26
2.6.1.	Test Data .....	26
2.6.2.	Quality Control .....	27
2.6.2.1.	Manual Error Handlers .....	27
2.6.2.2.	Pylint .....	28
2.6.2.3.	Flake8.....	29
2.6.2.4.	Mccabe .....	30
2.6.2.5.	Cyclomatic complexity .....	30
2.6.2.6.	Bandit .....	32
2.6.2.7.	Other tests.....	32
2.7.	Summary and Conclusion .....	34
2.8.	References .....	36

2.8.1. Reference list for the codes: .....	36
2.8.2. Reference list on security among others.....	36

## **Table of Figures**

Figure 1: Development server link .....	6
Figure 2: Production server link.....	7
Figure 3: Login page of repository .....	7
Figure 4: Security Features of the Repository .....	8
Figure 5: Bcrypt and hashed passwords .....	9
Figure 6: Flask-Hashing .....	10
Figure 7: Hashing and secure Download Button .....	11
Figure 8: Caesar Cypher Algorithm – trial 1.....	12
Figure 9: One-time pad algorithm – trial 1 .....	13
Figure 10: Reverse Cypher Algorithm – trial 1.....	13
Figure 11: Session Timeout updated.....	15
Figure 12: Environmental Configurations .....	15
Figure 13: Login Manager in Flask .....	16
Figure 14: Flask-Authorize.....	17
Figure 15: Regular Expressions .....	18
Figure 16: Email Validators instead of Regex.....	19
Figure 17: Secrets Keys with Random .....	20
Figure 18: OTP with Random (not secure) .....	20
Figure 19: Secrets used instead of Random.....	21
Figure 20: Wrong data input will raise errors .....	21
Figure 21: codes for Admin_dashboard.....	22
Figure 22: users' signing in and uploading a document.....	23
Figure 23: The user's activities on the admin dashboard (Activity log) .....	25
Figure 24: GDPR right to be forgotten .....	25
Figure 25: Manual error handling.....	27
Figure 26: Pylint.....	29
Figure 27: Flake8.....	29
Figure 28: Mccabe .....	30
Figure 29: Cyclomatic Complexity Testing.....	31
Figure 30: Cyclomatic Complexity Results Explanation.....	31
Figure 31: Bandit .....	32
Figure 32: Simple flask app .....	33
Figure 33: pytest codes .....	33
Figure 34: pytest results (in the command prompt) .....	34

## **Table of Tables**

Table 1: Work Plan .....	4
Table 2: An overview of the different cryptography algorithms .....	12
Table 3: Python libraries against vulnerabilities.....	26

## 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<sup>1</sup>

---

<sup>1</sup> 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

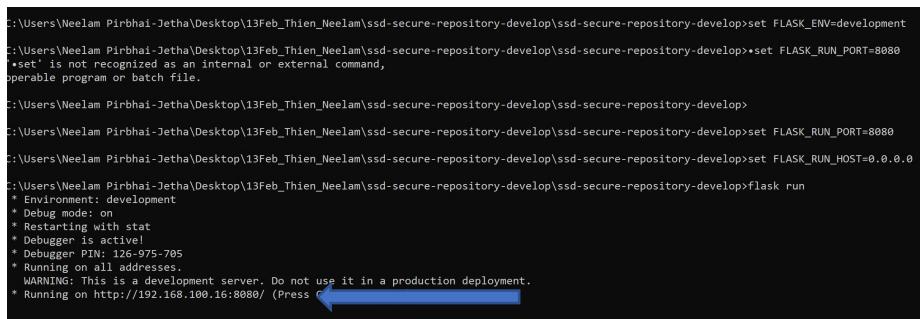
### On Windows:

- 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:



```
C:\Users\Neelam Pirbhai-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop>set FLASK_ENV=development
C:\Users\Neelam Pirbhai-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop>set FLASK_RUN_PORT=8080
*set' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\Neelam Pirbhai-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop>ssd-secure-repository-develop>set FLASK_RUN_PORT=8080
C:\Users\Neelam Pirbhai-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop>ssd-secure-repository-develop>set FLASK_RUN_HOST=0.0.0.0
C:\Users\Neelam Pirbhai-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop>ssd-secure-repository-develop>flask run
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active
 * Debugger PIN: 322-075-705
 * Running on all addresses.
 * WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://192.168.100.16:8080/ (Press Ctrl-C to quit)
```

Figure 1: Development server link

## The production environment is for deployment:

```
Command Prompt
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Neelam Pirbhais-Jetha>cd C:\Users\Neelam Pirbhais-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop\ssd-secure-repository-develop
C:\Users\Neelam Pirbhais-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop\ssd-secure-repository-develop>pip install -r requirements.txt

C:\Users\Neelam Pirbhais-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop\ssd-secure-repository-develop>python create_user.py
C:\Users\Neelam Pirbhais-Jetha\Desktop\13Feb_Thien_Neelam\ssd-secure-repository-develop\ssd-secure-repository-develop>flask run
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit) ←
```

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.

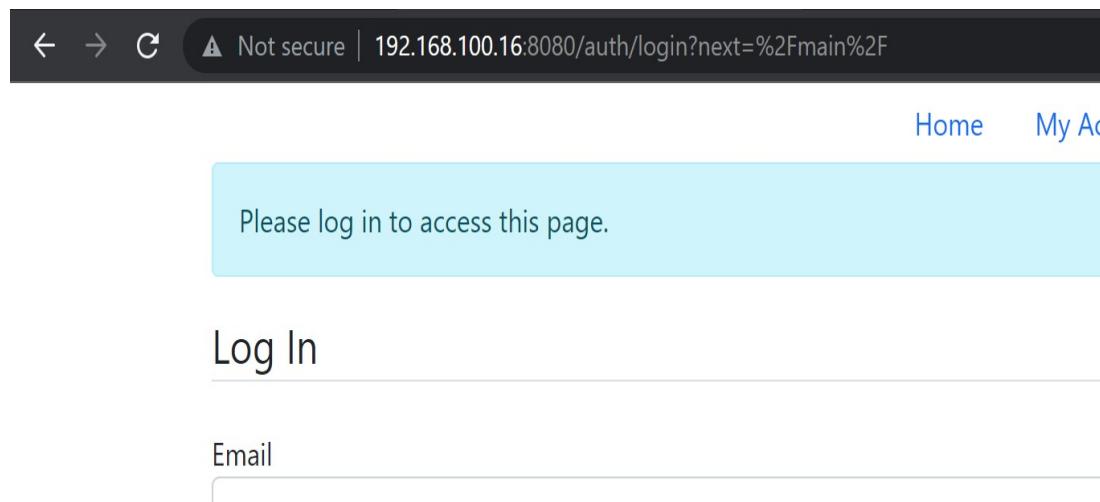
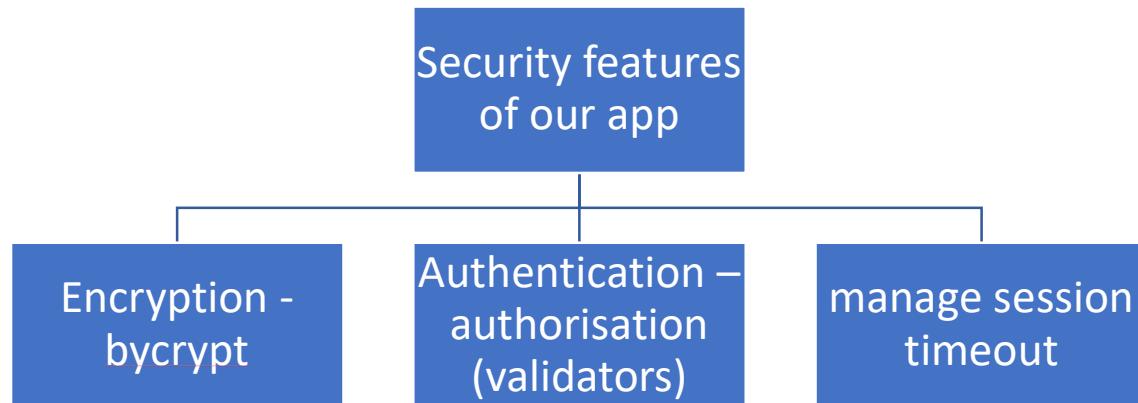


Figure 3: Login page of repository

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

```
11 """Flask-Bcrypt is a Flask extension that provides bcrypt hashing utilities:  
12 to protect passwords for instance  
13 Then below add 'Bcrypt='... ' to initialise it in the app; then go to route.py  
14 """  
15 from flask_bcrypt import Bcrypt
```

```
❷ create_user.py ✘  
❸ create_user.py > ⌂ main  
1 import email  
2 from app import db, bcrypt  
3 from app.auth.models import User  
4 import sys  
5  
6 def main():  
7     password = "Qwerty@123"  
8     hashed_password = bcrypt.generate_password_hash(password)  
9     user1 = User(  
10         name = "Thien",  
11         email = "thien@awesomerepo.com",  
12         password = hashed_password,  
13         role = 1,  
14         status = 2  
15     )  
16     user2 = User(  
17         name = "Neelam",  
18         email = "neelam@awesomerepo.com",  
19         password = hashed_password,  
20         role = 1,  
21         status = 2  
22     )  
23     user3 = User(  
24         name = "Created Internal Staff",  
25         email = "internal_created@awesomerepo.com",  
26         password = hashed_password,  
27         role = 2,  
28         status = 1  
29     )  
30     db.session.add(user1)  
31     db.session.add(user2)  
32     db.session.add(user3)
```

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.

The screenshot shows a Python code editor with the URL [flask-hashing.readthedocs.io/en/latest/](https://flask-hashing.readthedocs.io/en/latest/) in the address bar. The page title is "Flask-Hashing". A sidebar message says: "Even if we use these libraries for our development phase, there might be some changes (in progress)". The main content is a code listing for `FileService.py`:

```
 1  from app import db
 2  from app.document.models import File
 3  from werkzeug.datastructures import FileStorage
 4  from werkzeug.utils import secure_filename
 5  from hashlib import sha256
 6  from app.errors.filesError import (
 7      FileAlreadyExistsError,
 8      FileInsertionError,
 9  )
10
11 class FileService:
12
13     @classmethod
14         def get_user_files(cls, user_id: int):
15             files = db.session.query(File).filter_by(owner_id=user_id).order_by(
16                 File.created_at.desc()
17             )
18             return files
19
20     @classmethod
21         def create_file(cls, uploaded_file: FileStorage, user_id: int):
22             filename = secure_filename(uploaded_file.filename)
23             if db.session.query(File).filter_by(owner_id=user_id, title=filename).first() is not None:
24                 raise FileAlreadyExistsError(filename)
25             blob = uploaded_file.read()
26             # FileStorage class (which is the class to handle uploaded file in Flask)
27             # points to end of file after every action (saving or reading).
28             uploaded_file.stream.seek(0)
29             size = len(blob)
30             f_hash = sha256(blob).hexdigest()
```

The code is syntax-highlighted with numbers on the left. A status bar at the bottom right shows "English (United States) US".

Figure 6: Flask-Hashing

### 2.5.1.3. Encrypting documents

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

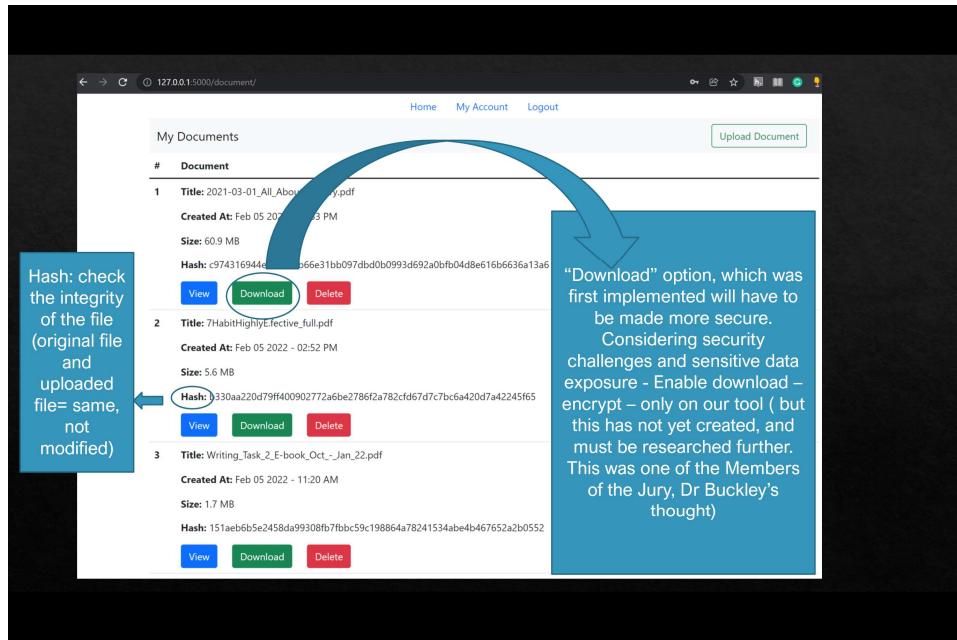
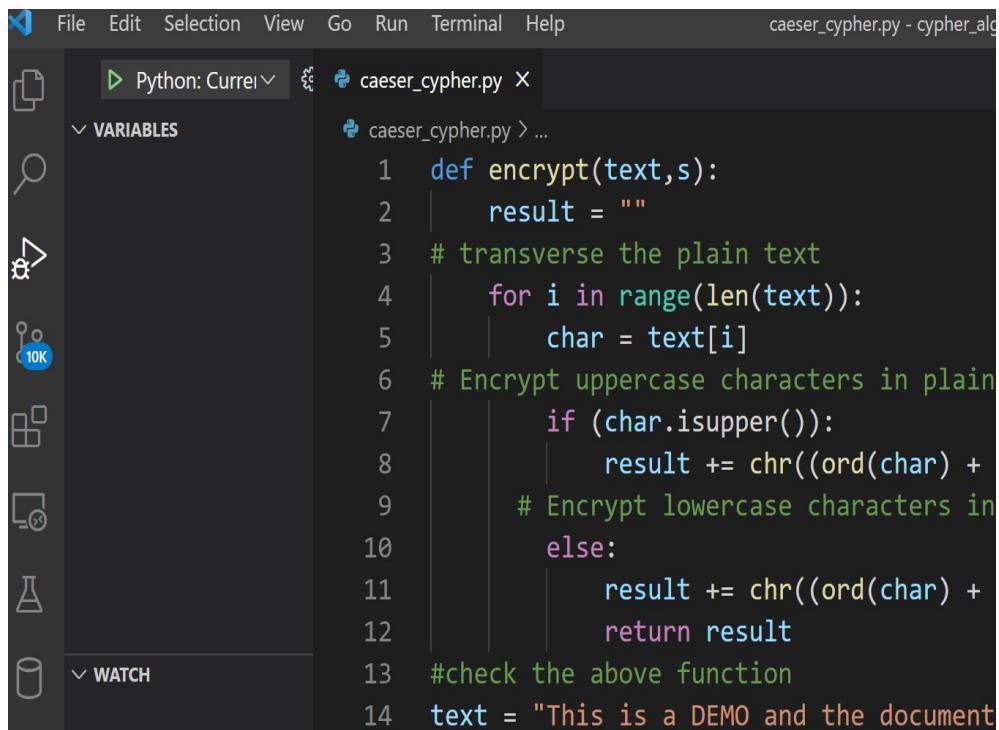


Figure 7: Hashing and secure Download Button

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 cipher 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



The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The current file is "caeser\_cypher.py". The code editor displays the following Python script:

```

File Edit Selection View Go Run Terminal Help
caeser_cypher.py - cypher_alg

Python: Current caeser_cypher.py
VARIABLES caeser_cypher.py > ...
1 def encrypt(text,s):
2     result = ""
3     # transverse the plain text
4     for i in range(len(text)):
5         char = text[i]
6         # Encrypt uppercase characters in plain
7         if (char.isupper()):
8             result += chr((ord(char) +
9                           s)%256)
10            # Encrypt lowercase characters in plain
11            else:
12                result += chr((ord(char) +
13                               s)%256)
14    #check the above function
15    text = "This is a DEMO and the document"

```

The sidebar on the left contains icons for file operations like Open, Save, Find, and Run, along with a "10K" icon.

Figure 8: Caesar Cypher Algorithm – trial 1

A screenshot of a Python code editor interface. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The current file is "onetimepad\_cypher.py". The code window shows the following Python script:

```
1 import onetimepad
2
3 cipher = onetimepad.encrypt('This is a
4 print("Cipher text is: ")
5 print(cipher)
6 print("Plain text is: ")
7 msg = onetimepad.decrypt(cipher, 'random'
8
9 print(msg)
```

Figure 9: One-time pad algorithm – trial 1

A screenshot of a Python code editor interface. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The current file is "reverse\_cypher.py". The code window shows the following Python script:

```
1 #reverse cypher
2 message = input ("Add your text here:")
3 print(message)
4 translated = " " #cipher text is stored
5 i = len(message) - 1
6
7 while i >= 0:
8     translated = translated + message[i]
9     i = i - 1
10 print("The cipher text is : ", translated)
```

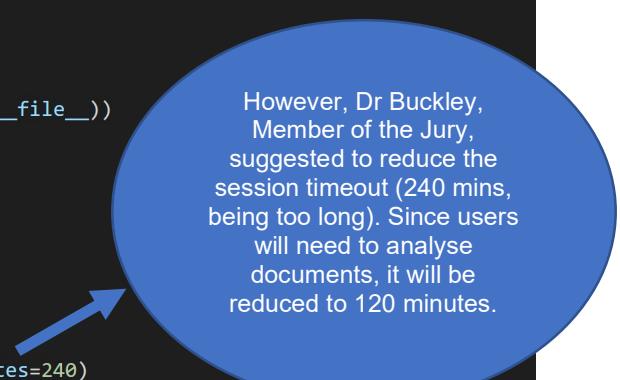
Figure 10: Reverse Cypher Algorithm – trial 1

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.

```
config.py > TestingConfig
1 import os
2 from datetime import timedelta
3
4 BASE_DIR = os.path.abspath(os.path.dirname(__file__))
5 class Config(object):
6     DEBUG = True
7     TESTING = False
8     SQLALCHEMY_TRACK_MODIFICATIONS = False
9     CSRF_ENABLED = True
10    CSRF_SESSION_KEY = os.urandom(24)
11
12    # Session
13    PERMANENT_SESSION_LIFETIME = timedelta(minutes=240)
14
15    class DevelopmentConfig(Config):
16        SECRET_KEY = os.getenv('SECRET_KEY_DEV', os.urandom(24))
17        DATABASE_NAME = "awesome_repo_dev.db"
18        SQLALCHEMY_DATABASE_URI = 'sqlite:///{}' + os.path.join(BASE_DIR, DATABASE_NAME)
19        ADMINDASHBOARD_VISIBLE = True
20    class TestingConfig(Config):
21        SECRET_KEY = os.getenv('SECRET_KEY_TEST', os.urandom(24))
22        DATABASE_NAME = "awesome_repo_test.db"
23        SQLALCHEMY_DATABASE_URI = 'sqlite:///{}' + os.path.join(BASE_DIR, DATABASE_NAME)
24        ADMINDASHBOARD_VISIBLE = True
25    class ProductionConfig(Config):
26        SECRET_KEY = os.getenv('SECRET_KEY_PROD', os.urandom(24))
27        DATABASE_NAME = "awesome_repo_prod.db"
28        SQLALCHEMY_DATABASE_URI = 'sqlite:///{}' + os.path.join(BASE_DIR, DATABASE_NAME)
29        ADMINDASHBOARD_VISIBLE = False
30
31 config = {
```



However, Dr Buckley, Member of the Jury, suggested to reduce the session timeout (240 mins, being too long). Since users will need to analyse documents, it will be reduced to 120 minutes.

```

12
13 # Session
14 PERMANENT_SESSION_LIFETIME = timedelta(minutes=120)
15

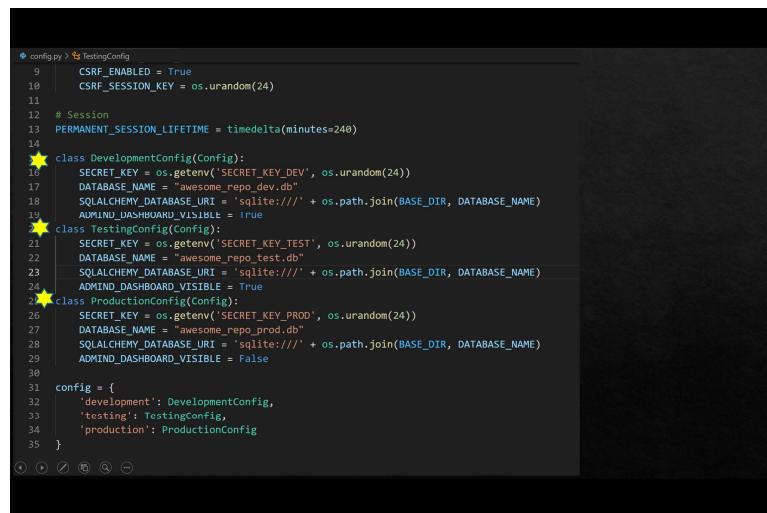
```



*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) (Aramatzis, 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).



```

config.py > 4 TestingConfig
9     CSRF_ENABLED = True
10    CSRF_SESSION_KEY = os.urandom(24)
11
12 # Session
13 PERMANENT_SESSION_LIFETIME = timedelta(minutes=240)
14
15 class DevelopmentConfig(Config):
16     SECRET_KEY = os.getenv('SECRET_KEY_DEV', os.urandom(24))
17     DATABASE_NAME = "awesome_repo.dev.db"
18     SQLALCHEMY_DATABASE_URI = 'sqlite:///{} + os.path.join(BASE_DIR, DATABASE_NAME)
19     ADMINDASHBOARD_VISIBLE = True
20
21 class TestingConfig(Config):
22     SECRET_KEY = os.getenv('SECRET_KEY_TEST', os.urandom(24))
23     DATABASE_NAME = "awesome_repo.test.db"
24     SQLALCHEMY_DATABASE_URI = 'sqlite:///{} + os.path.join(BASE_DIR, DATABASE_NAME)
25     ADMINDASHBOARD_VISIBLE = True
26
27 class ProductionConfig(Config):
28     SECRET_KEY = os.getenv('SECRET_KEY_PROD', os.urandom(24))
29     DATABASE_NAME = "awesome_repo.prod.db"
30     SQLALCHEMY_DATABASE_URI = 'sqlite:///{} + os.path.join(BASE_DIR, DATABASE_NAME)
31     ADMINDASHBOARD_VISIBLE = False
32
33 config = {
34     'development': DevelopmentConfig,
35     'testing': TestingConfig,
36     'production': ProductionConfig
37 }

```

*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.

```
17 """
18 pip install flask-login in cmd
19 then import LoginManager
20 "Flask-Login provides user session management for Flask.
21 It handles the common tasks of logging in, logging out,
22 and remembering your users' sessions over extended periods of time." [Flask Login Docume
23 """
24 from flask_login import LoginManager
25

16
17 # Import the database object from the main app module
18 from app import db, bcrypt, login_manager
19
20 from flask_login import login_user, current_user, logout_user, login_required, LoginManager
21
22 from app.auth.forms import LoginForm, ChangePasswordForm
23 from app.auth.models import User
24
25 auth = Blueprint('auth', __name__, url_prefix='/auth')
26 app_config = current_app.config
27
28 # A user_loader callback, used to reload the user object
29 # from the user ID stored in the session
30 @login_manager.user_loader
31 def load_user(user_id):
32     return User.query.get(int(user_id))
33
34 @auth.route('/login', methods=['GET', 'POST'])
35 def login():
36     if current_user.is_authenticated:
37         if current_user.isAdmin:
38             return redirect(url_for('admin.home'))
39         else:
40             return redirect(url_for('main.home'))
41
42     form = LoginForm(request.form)
43
44     if form.validate_on_submit():
45         user = User.query.filter_by(email=form.email.data).first()
46         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).



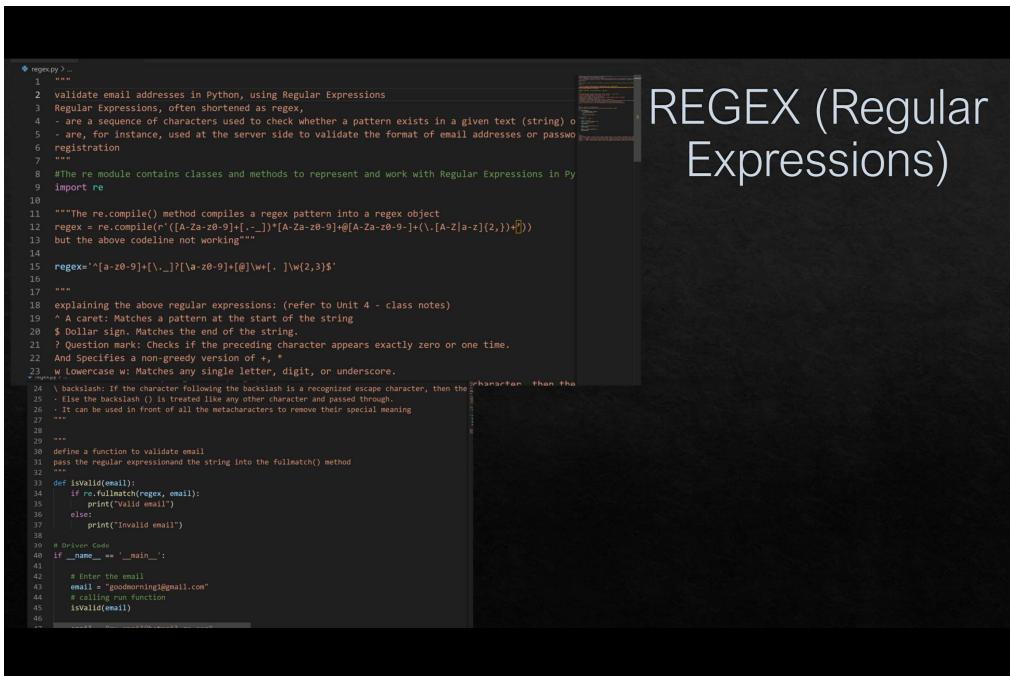
The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "SSD-SECURE-REPOSITORY-...". Key files visible include `app`, `admin`, `auth`, `models.py`, `run.py`, and configuration files like `.gitignore`, `Pipfile`, and `README.md`.
- models.py** file content (line numbers 5-34):

```
5 from flask_authorize import RestrictionsMixin, AllowancesMixin
6
7
8 #mapping tables
9 UserGroup = db.Table(
10     'user_group', db.Model.metadata,
11     db.Column('user_id', db.Integer, db.ForeignKey('users.id')),
12     db.Column('group_id', db.Integer, db.ForeignKey('groups.id'))
13 )
14
15 UserRole = db.Table(
16     'user_role', db.Model.metadata,
17     db.Column('user_id', db.Integer, db.ForeignKey('users.id')),
18     db.Column('role_id', db.Integer, db.ForeignKey('roles.id'))
19 )
20
21 class Group(db.Model, RestrictionsMixin):
22     __tablename__ = 'groups'
23
24     id = db.Column(db.Integer, primary_key=True)
25     name = db.Column(db.String(255), nullable=False, unique=True)
26
27     # Authorization Data: role & status
28     # role:
29     # 1: admin
30     # 2: internal
31     # 3: external
32
33     class Role(db.Model, AllowancesMixin):
34         __tablename__ = 'roles'
```

Figure 14: Flask-Authorize

## 2.5.5. Regex

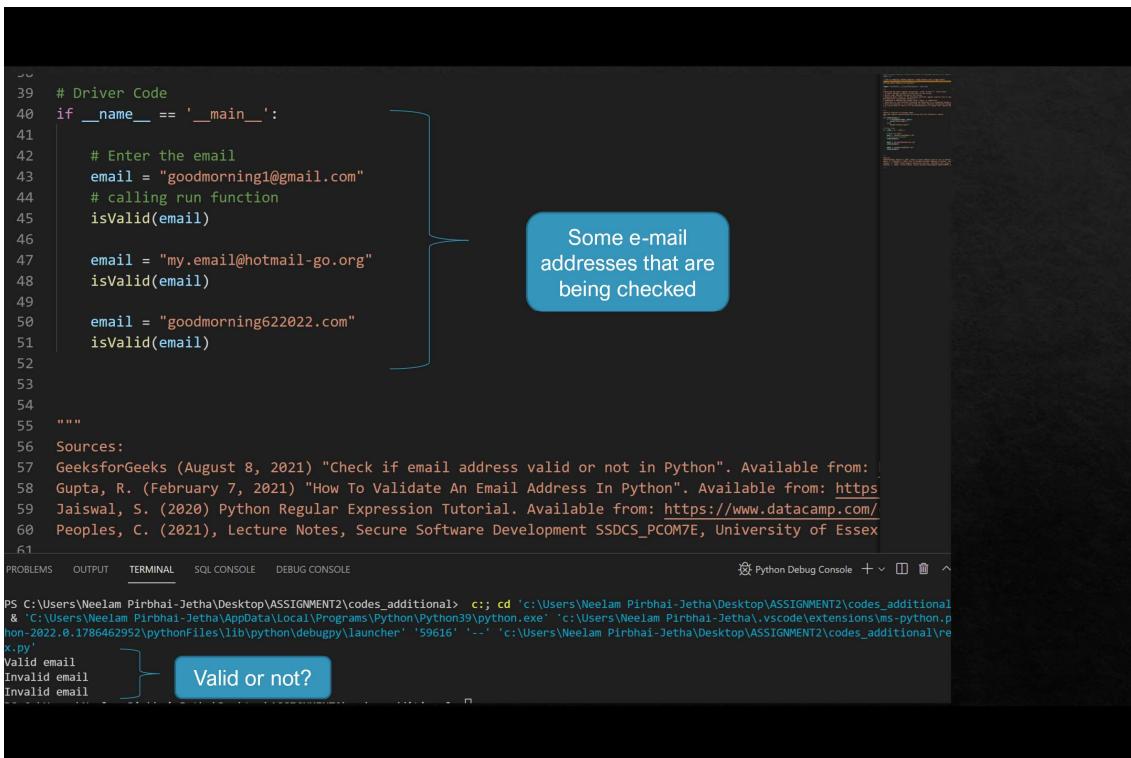


```

1 # regex.py ...
2 """
3 validate email addresses in Python, using Regular Expressions
4 Regular Expressions, often shortened as regex,
5 - are a sequence of characters used to check whether a pattern exists in a given text (string)
6 - are, for instance, used at the server side to validate the format of email addresses or passwords
7 registration
8
9 #The re module contains classes and methods to represent and work with Regular Expressions in Python
10 import re
11 """
12 """The re.compile() method compiles a regex pattern into a regex object
13 regex = re.compile(r'^([A-Za-z0-9]+[._-])*[A-Za-z0-9-]+@[A-Za-z0-9-]+\.(.[A-Z|a-z]{2,})+$')
14 but the above codeline not working"""
15
16 regexes="^([A-Za-z0-9]+[._-])?([A-Za-z0-9-]+@[A-Za-z0-9-]+\.(.[A-Z|a-z]{2,})$"
17 """
18 explaining the above regular expressions: (refer to Unit 4 - class notes)
19 ^ A caret: Matches a pattern at the start of the string
20 $ Dollar sign: Matches the end of the string.
21 ? Question mark: Checks if the preceding character appears exactly zero or one time.
22 And Specifies a non-greedy version of *,
23 . Lowercase w: Matches any single letter, digit, or underscore.
24 \ Backslash: Escapes characters following the backslash as a recognized escape character, then the character than the
25 backslash() is treated as a regular character and passed through.
26 . It can be used in front of all the metacharacters to remove their special meaning
27 """
28 """
29 """
30 define a function to validate email
31 pass the regular expression and the string into the fullmatch() method
32 """
33 def isValid(email):
34     if re.fullmatch(regex, email):
35         print("Valid email")
36     else:
37         print("Invalid email")
38
39 # Driver Code
40 if __name__ == '__main__':
41
42     # Enter the email
43     email = "goodmorning1@gmail.com"
44     # calling run function
45     isValid(email)
46
47     email = "my.email@hotmail-go.org"
48     isValid(email)
49
50     email = "goodmorning622022.com"
51     isValid(email)
52
53
54 """
55 Sources:
56 GeeksforGeeks (August 8, 2021) "Check if email address valid or not in Python". Available from:
57 Gupta, R. (February 7, 2021) "How To Validate An Email Address In Python". Available from: https://www.geeksforgeeks.org/check-if-email-address-valid-or-not-in-python/
58 Jaiswal, S. (2020) Python Regular Expression Tutorial. Available from: https://www.datacamp.com/
59 Peoples, C. (2021), Lecture Notes, Secure Software Development SSDCS_PCOM7E, University of Essex
60

```

## REGEX (Regular Expressions)



```

1 # Driver Code
2 if __name__ == '__main__':
3
4     # Enter the email
5     email = "goodmorning1@gmail.com"
6     # calling run function
7     isValid(email)
8
9     email = "my.email@hotmail-go.org"
10    isValid(email)
11
12    email = "goodmorning622022.com"
13    isValid(email)
14
15
16 """
17 Sources:
18 GeeksforGeeks (August 8, 2021) "Check if email address valid or not in Python". Available from:
19 Gupta, R. (February 7, 2021) "How To Validate An Email Address In Python". Available from: https://www.geeksforgeeks.org/check-if-email-address-valid-or-not-in-python/
20 Jaiswal, S. (2020) Python Regular Expression Tutorial. Available from: https://www.datacamp.com/
21 Peoples, C. (2021), Lecture Notes, Secure Software Development SSDCS_PCOM7E, University of Essex
22

```

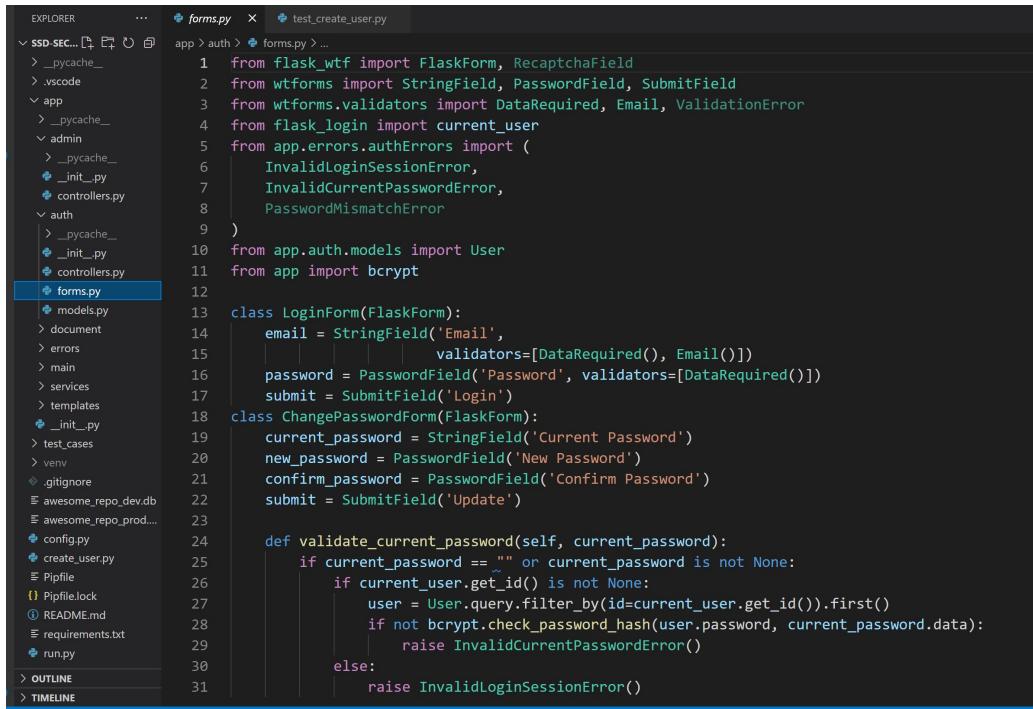
Some e-mail addresses that are being checked

Valid or not?

Valid email  
Invalid email  
Invalid email

Figure 15: Regular Expressions

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



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like `app`, `auth`, `models.py`, `run.py`, etc.
- forms.py** tab: Active file, showing Python code for Flask forms.
- test\_create\_user.py**: Another file tab in the background.
- Code Content:**

```
1  from flask_wtf import FlaskForm, RecaptchaField
2  from wtforms import StringField, PasswordField, SubmitField
3  from wtforms.validators import DataRequired, Email, ValidationError
4  from flask_login import current_user
5  from app.errors.authErrors import (
6      InvalidLoginSessionError,
7      InvalidCurrentPasswordError,
8      PasswordMismatchError
9  )
10 from app.auth.models import User
11 from app import bcrypt
12
13 class LoginForm(FlaskForm):
14     email = StringField('Email',
15                         validators=[DataRequired(), Email()])
16     password = PasswordField('Password', validators=[DataRequired()])
17     submit = SubmitField('Login')
18
19 class ChangePasswordForm(FlaskForm):
20     current_password = StringField('Current Password')
21     new_password = PasswordField('New Password')
22     confirm_password = PasswordField('Confirm Password')
23     submit = SubmitField('Update')
24
25     def validate_current_password(self, current_password):
26         if current_password == "" or current_password is not None:
27             if current_user.get_id() is not None:
28                 user = User.query.filter_by(id=current_user.get_id()).first()
29                 if not bcrypt.check_password_hash(user.password, current_password.data):
30                     raise InvalidCurrentPasswordError()
31             else:
32                 raise InvalidLoginSessionError()
```

Figure 16: Email Validators instead of Regex

## 2.5.6. Generating Secret Keys

```
◆ config.py ×
◆ config.py > TestingConfig
1 import os
2 from datetime import timedelta
3
4 BASE_DIR = os.path.abspath(os.path.dirname(__file__))
5 class Config(object):
6     DEBUG = True
7     TESTING = False
8     SQLALCHEMY_TRACK_MODIFICATIONS = False
9     CSRF_ENABLED = True
10    CSRF_SESSION_KEY = os.urandom(24)
11
12 # Session
13 PERMANENT_SESSION_LIFETIME = timedelta(minutes=240)
14
15 class DevelopmentConfig(Config):
16     SECRET_KEY = os.getenv('SECRET_KEY_DEV', os.urandom(24))
17     DATABASE_NAME = "awesome_repo_dev.db"
18     SQLALCHEMY_DATABASE_URI = 'sqlite:///{} + os.path.join(BASE_DIR, DATABASE_NAME)'
19     ADMINDASHBOARD_VISIBLE = True
20
21 class TestingConfig(Config):
22     SECRET_KEY = os.getenv('SECRET_KEY_TEST', os.urandom(24))
23     DATABASE_NAME = "awesome_repo_test.db"
24     SQLALCHEMY_DATABASE_URI = 'sqlite:///{} + os.path.join(BASE_DIR, DATABASE_NAME)'
25     ADMINDASHBOARD_VISIBLE = True
26
27 class ProductionConfig(Config):
28     SECRET_KEY = os.getenv('SECRET_KEY_PROD', os.urandom(24))
29     DATABASE_NAME = "awesome_repo_prod.db"
30     SQLALCHEMY_DATABASE_URI = 'sqlite:///{} + os.path.join(BASE_DIR, DATABASE_NAME)'
31     ADMINDASHBOARD_VISIBLE = False
32
33 config = {
34     'development': DevelopmentConfig,
35     'testing': TestingConfig,
36     'production': ProductionConfig
37 }
```

*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:

The screenshot shows a code editor with a dark theme. On the left is a file tree for a repository named 'SSD-SECURE-REPOSITORY...'. The 'config.py' file is selected. The code in 'config.py' uses the 'secrets' module to generate a secure session key:

```
import os
import secrets
from datetime import timedelta

BASE_DIR = os.path.abspath(os.path.dirname(__file__))
class Config(object):
    DEBUG = True
    TESTING = False
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    CSRF_ENABLED = True
    CSRF_SESSION_KEY = secrets.token_urlsafe(24)

    # Session
    PERMANENT_SESSION_LIFETIME = timedelta(minutes=120)
```

Figure 19: Secrets used instead of Random

### 2.5.7. Raising errors

The screenshot shows a code editor with a dark theme. It displays two files: 'filesError.py' and 'authErrors.py'. 'filesError.py' contains classes for FileAlreadyExistsError, FileInsertionError, FileUpdateError, and FileDeletionError. 'authErrors.py' contains classes for InvalidLoginSessionError, InvalidCurrentPasswordError, and PasswordMismatchError. A callout bubble points to the error classes with the text: 'Codes written to raise errors if input of data is wrong'.

```
class FileAlreadyExistsError(Exception):
    def __init__(self, filename: str):
        self.filename = filename
        super().__init__()

class FileInsertionError(Exception):
    def __init__(self, filename: str):
        self.filename = filename
        super().__init__()

class FileUpdateError(Exception):
    def __init__(self, filename: str):
        self.filename = filename
        super().__init__()

class FileDeletionError(Exception):
    def __init__(self, filename: str):
        self.filename = filename
        super().__init__()

class InvalidLoginSessionError(Exception):
    def __init__(self):
        super().__init__()

class InvalidCurrentPasswordError(Exception):
    def __init__(self):
        super().__init__()

class PasswordMismatchError(Exception):
    def __init__(self):
        super().__init__()
```

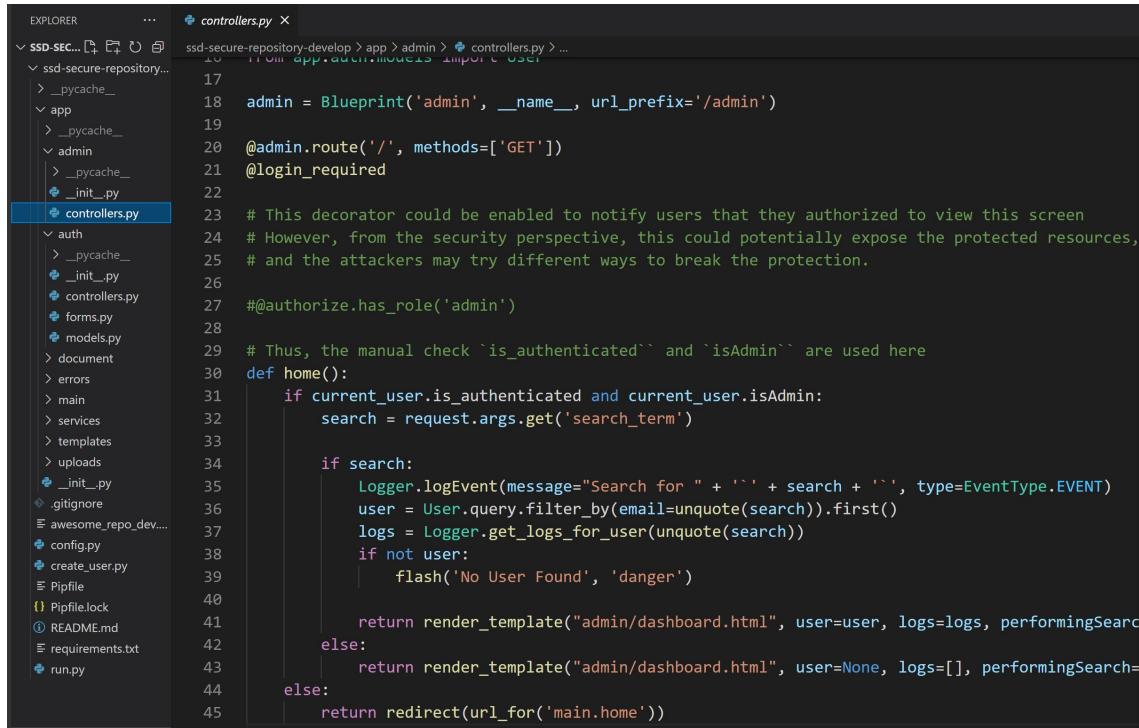
Figure 20: Wrong data input will raise errors

### 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)



The screenshot shows a code editor with the file 'controllers.py' open. The code defines an 'admin' blueprint with a route for the root path ('/') that requires login. It includes a manual check for user authentication and admin status. It also handles a search function by logging events and returning results if a user is found, or a 'No User Found' message if not. Finally, it renders an 'admin/dashboard.html' template.

```
EXPLORER ... controllers.py
ssd-secure-repository... ssd-secure-repository-develop > app > admin > controllers.py > ...
17
18     admin = Blueprint('admin', __name__, url_prefix='/admin')
19
20     @admin.route('/', methods=['GET'])
21     @login_required
22
23     # This decorator could be enabled to notify users that they authorized to view this screen
24     # However, from the security perspective, this could potentially expose the protected resources,
25     # and the attackers may try different ways to break the protection.
26
27     #@authorize.has_role('admin')
28
29     # Thus, the manual check `is_authenticated` and `isAdmin` are used here
30     def home():
31         if current_user.is_authenticated and current_user.isAdmin:
32             search = request.args.get('search_term')
33
34             if search:
35                 Logger.logEvent(message="Search for " + search, type=EventType.EVENT)
36                 user = User.query.filter_by(email=unquote(search)).first()
37                 logs = Logger.get_logs_for_user(unquote(search))
38                 if not user:
39                     flash('No User Found', 'danger')
40
41                     return render_template("admin/dashboard.html", user=user, logs=logs, performingSearch=True)
42                 else:
43                     return render_template("admin/dashboard.html", user=None, logs=[], performingSearch=False)
44             else:
45                 return redirect(url_for('main.home'))
```

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

Home My Account Logout

Please log in to access this page.

Log In

Email  
internal\_created@awesomerepo.com

Password  
\*\*\*\*\*

Login

Then upload any document:

Not secure | 192.168.100.16:8080/document/

Home

My Documents

Not secure | 192.168.100.16:8080/document/upload

Home

Select a file to upload

Not secure | 192.168.100.16:8080/document/

Home

My Documents

#	Document
1	Title: 10Feb_Development_Team_Project.pptx

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

Admin

[Home](#)   [My Account](#)   [Logout](#)

Please log in to access this page.

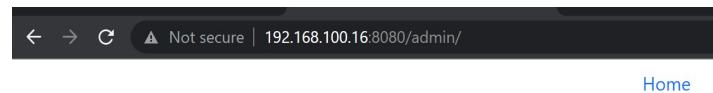
Log In

Email

Password

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:

The screenshot shows a web browser window with the URL `192.168.100.16:8080/admin/?search_term=internal_created%40awesomerepo.com`. The page title is "Admin dashboard". A search bar contains the placeholder "Search User By Email". Below it, a message says "Found User: internal\_created@awesomerepo.com". A table titled "# Activity" lists one event: "1 Event Type: EVENT" with the message "Message: View Documents".

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

#### 2.5.9. GDPR 'right to be forgotten'

The screenshot shows the same admin dashboard as Figure 23. A large blue arrow points from the right side of the screen towards the back button in the browser's address bar.

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 detail 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	<ul style="list-style-type: none"><li>Flask-authorize</li></ul>
Sensitive data exposure	<ul style="list-style-type: none"><li>Hashlib</li><li>Bcrypt</li></ul>
Denial of Service (Erickson)	<ul style="list-style-type: none"><li>Don't have the configs yet</li><li>API rate limit – too many requests will raise an error</li></ul>
Injection	<ul style="list-style-type: none"><li>email-validators</li></ul>
Buffer overflow	<ul style="list-style-type: none"><li>Investigate Flask-authorize, Bcrypt, etc. and check if there is no buffer-overflow in the existing library</li></ul>

*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

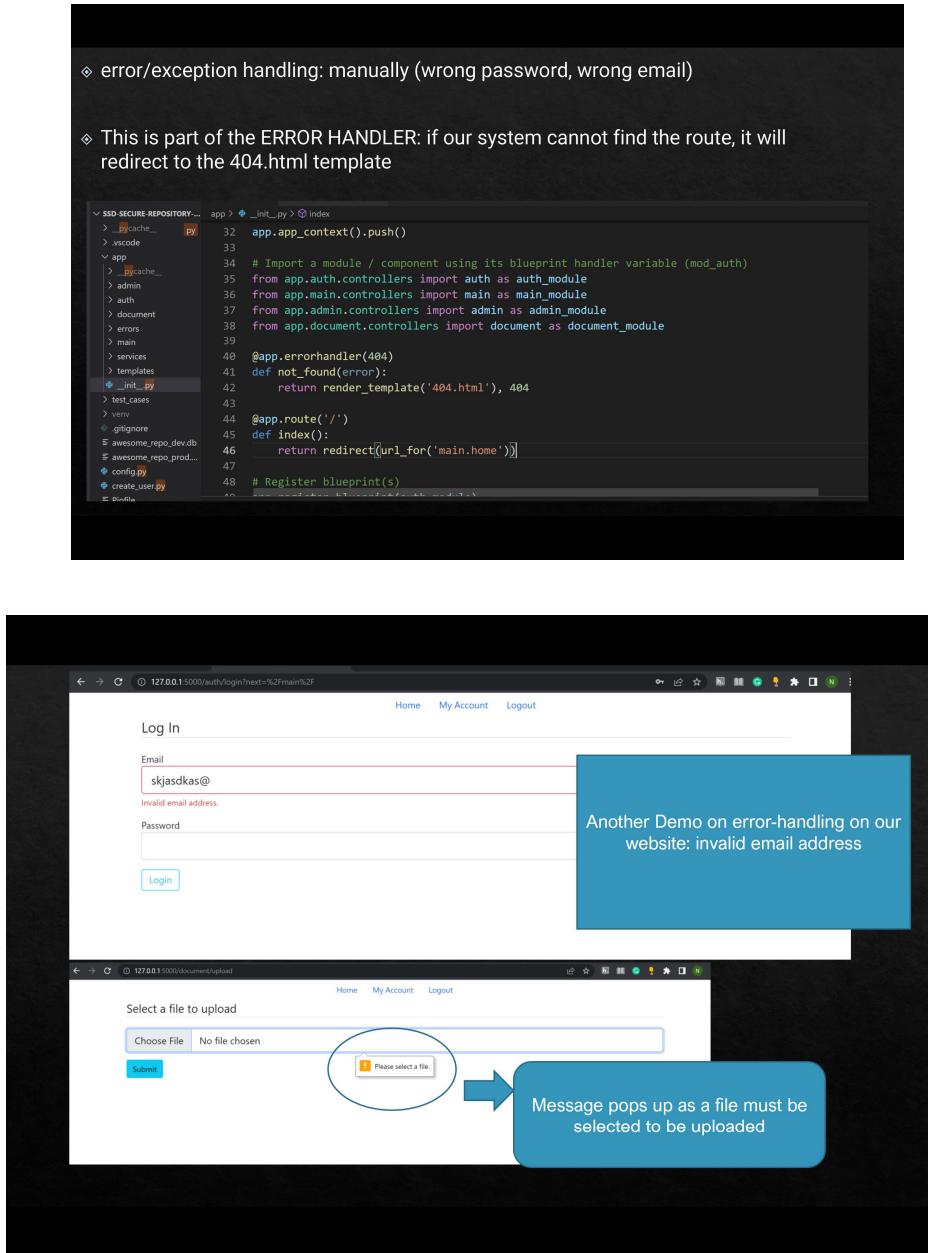


Figure 25: Manual error handling

### 2.6.2.2. Pylint

The screenshot shows a Jupyter Notebook interface with a code cell containing Python code for generating OTPs using the secrets module. The code includes user input for a username and an OTP, and prints a message with the OTP. A large blue callout bubble points from the right towards the terminal window, containing the text "OTP will be added later to our codes".

```
◆ secure_otp.py > ...
1 """
2 Generate a secure secret code
3 Instead of random, use secrets module
4 """
5
6 import secrets
7 # Getting systemRandom class instance out of secrets module
8
9 username = input("Enter Username: ")
10
11 otp = secrets.token_hex(16)
12 print("Hello " + username + "!" + " Your OTP is " + otp)
13
14 password=input("Enter the OTP:")
15 if password==otp:
16     print("OTP is correct.")
17 else:
18     print("OTP is incorrect.")
19
20
?1
```

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE

```
PS C:\Users\Neelam Pirbhail-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:2:10: C0305: Trailing newline(s) (trailing-newlines)

Your code has been rated at 7.50/10 (previous run: 7.50/10, +0.00)

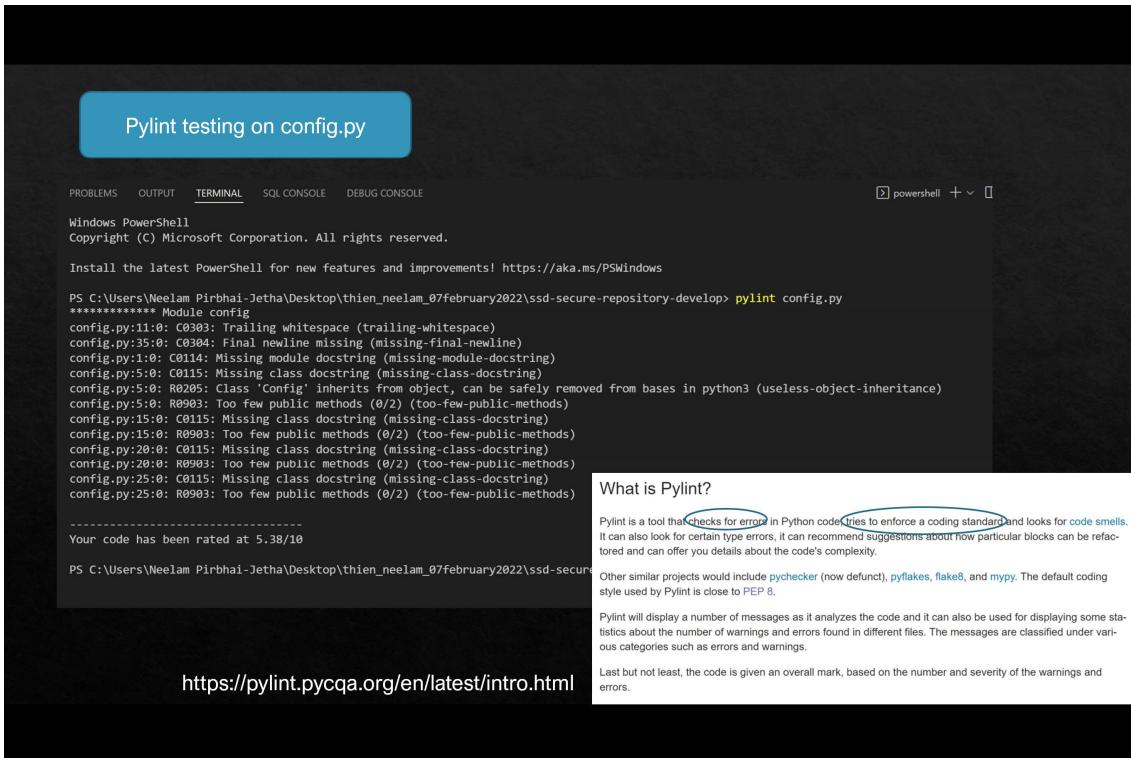
PS C:\Users\Neelam Pirbhail-Jetha\Desktop\ASSIGNMENT2\codes_additional> ■
```

In the Terminal: pylint file\_name.py

## Testing: pylint

OTP will be added later to our codes

In the Terminal: `pylint file_name.py`



## 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.

```

Pylint testing on create_user.py

PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> pylint create_user.py
*****
Module create_user
-----
create_user.py:11:40: C0303: Trailing whitespace (trailing-whitespace)
create_user.py:13:17: C0303: Trailing whitespace (trailing-whitespace)
create_user.py:18:41: C0303: Trailing whitespace (trailing-whitespace)
create_user.py:20:17: C0303: Trailing whitespace (trailing-whitespace)
create_user.py:25:51: C0303: Trailing whitespace (trailing-whitespace)
create_user.py:27:17: C0303: Trailing whitespace (trailing-whitespace)
create_user.py:30:1: E0401: Missing module docstring (missing-module-docstring)
create_user.py:6:0: C0116: Missing function or method docstring (missing-function-docstring)
create_user.py:30:4: E1101: Instance of 'scoped_session' has no 'add' member (no-member)
create_user.py:31:4: E1101: Instance of 'scoped_session' has no 'commit' member (no-member)
create_user.py:32:4: E1101: Instance of 'scoped_session' has no 'add' member (no-member)
create_user.py:33:4: E1101: Instance of 'scoped_session' has no 'commit' member (no-member)
create_user.py:1:0: W0611: Unused import email (unused-import)
create_user.py:4:0: C0411: standard import "import sys" should be placed before "from app import db, bcrypt" (wrong-import-order)

-----
Your code has been rated at -8.75/10

PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>

```

Figure 26: Pylint

### 2.6.2.3. Flake8

```

Testing flake8

PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 config.py
config.py:11:1: E301 expected 2 blank lines, found 0
config.py:11:1: W293 blank line contains whitespace
config.py:13:1: E301 expected 2 blank lines after class or function definition, found 1
config.py:18:1: E301 line too long (82 > 79 characters)
config.py:18:80: E501 line too long (82 > 79 characters)
config.py:20:1: E302 expected 2 blank lines, found 0
config.py:21:1: E301 expected 2 blank lines after class or function definition
config.py:25:1: E302 expected 2 blank lines, found 0
config.py:28:80: E501 line too long (82 > 79 characters)
config.py:31:1: E301 expected 2 blank lines after class or function definition, found 1
config.py:31:1: E301 line too long (82 > 79 characters)
PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>

PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 create_user.py
create_user.py:1:1: F401 'email' imported but unused
-----
create_user.py:6:1: E302 expected 2 blank lines, found 1
create_user.py:10:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:10:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:11:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:11:16: E251 unexpected spaces around keyword / parameter equals
create_user.py:11:41: W291 trailing whitespace
create_user.py:12:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:12:18: E251 unexpected spaces around keyword / parameter equals
create_user.py:13:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:13:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:13:18: W291 trailing whitespace
create_user.py:14:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:14:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:17:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:17:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:18:14: E251 unexpected spaces around keyword / parameter equals
create_user.py:18:18: E251 unexpected spaces around keyword / parameter equals
create_user.py:19:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:19:18: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:18: E251 unexpected spaces around keyword / parameter equals
create_user.py:21:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:21:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:24:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:24:18: E251 unexpected spaces around keyword / parameter equals
create_user.py:25:14: E251 unexpected spaces around keyword / parameter equals
create_user.py:25:18: W291 trailing whitespace
create_user.py:25:52: W291 trailing whitespace
create_user.py:26:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:26:19: E251 unexpected spaces around keyword / parameter equals
create_user.py:26:23: E251 unexpected spaces around keyword / parameter equals
create_user.py:27:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:27:18: W291 trailing whitespace
create_user.py:28:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:28:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:30:1: E305 expected 2 blank lines after class or function definition, found 1
PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>

```

Figure 27: Flake8

#### 2.6.2.4. McCabe

```

Testing McCabe
flake8 --max-complexity 10 name_of_file.py

PS C:\Users\Weelam\Pirbhali-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 --max-complexity 10 create_user.py
PS C:\Users\Weelam\Pirbhali-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 --max-complexity 10 config.py

```

The terminal window displays the output of the McCabe complexity analysis for two Python files: `create_user.py` and `config.py`. The results show numerous errors (E001) indicating complex code structures exceeding the specified complexity limit of 10. The errors are categorized by line number and type, such as E302 for unexpected spaces around keywords and E305 for unexpected spaces around operators.

Figure 28: McCabe

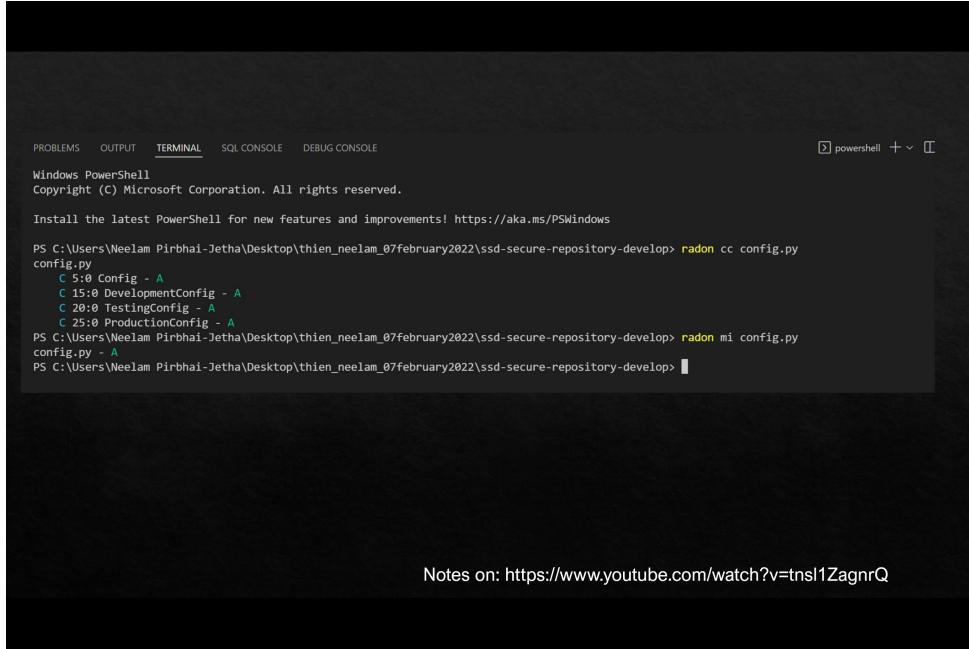
#### 2.6.2.5. Cyclomatic complexity

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)



The screenshot shows a Windows PowerShell window with the title bar "powershell". The terminal tab is active. The command `radon cc config.py` is run, followed by `radon mi config.py`. The output shows cyclomatic complexity scores for various configurations:

```

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

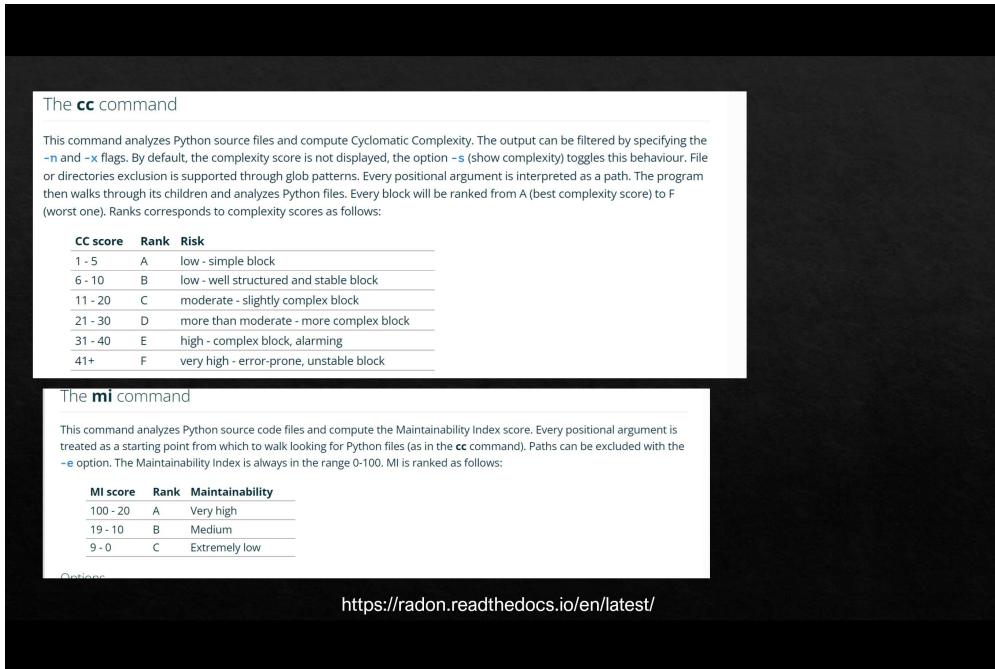
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> radon cc config.py
C 5:0 Config - A
C 15:0 DevelopmentConfig - A
C 20:0 TestingConfig - A
C 25:0 ProductionConfig - A
PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> radon mi config.py
config.py - A
PS C:\Users\Neelam Pirbhail-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>

```

Notes on: <https://www.youtube.com/watch?v=tnsl1ZagnrQ>

*Figure 29: Cyclomatic Complexity Testing*



The screenshot shows a section of the Radon documentation titled "The cc command". It explains that the command analyzes Python source files and computes Cyclomatic Complexity. It includes a table mapping complexity scores to ranks:

CC score	Rank	Risk
1 - 5	A	low - simple block
6 - 10	B	low - well structured and stable block
11 - 20	C	moderate - slightly complex block
21 - 30	D	more than moderate - more complex block
31 - 40	E	high - complex block, alarming
41+	F	very high - error-prone, unstable block

The screenshot also shows the "The mi command" section, which explains that it analyzes Python source code files and computes the Maintainability Index score. It includes a table mapping MI scores to ranks:

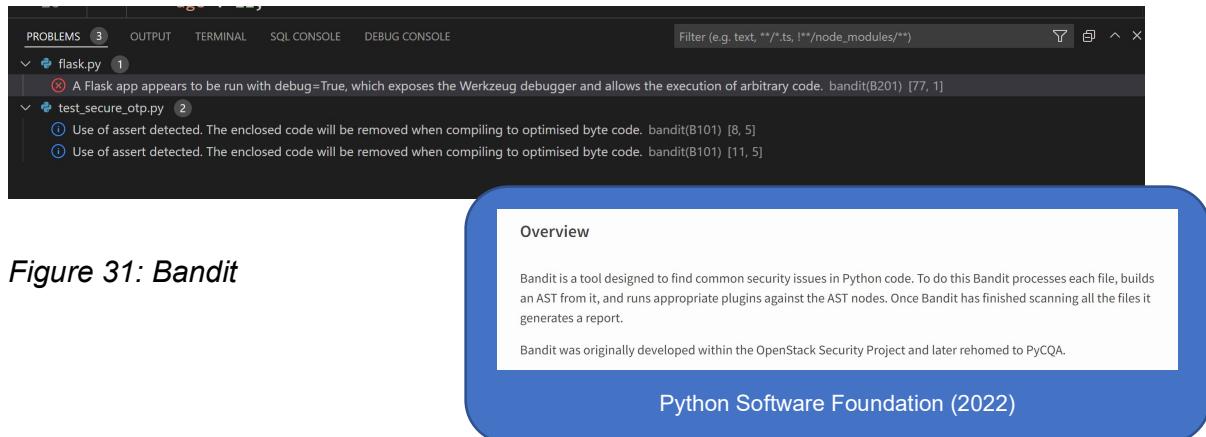
MI score	Rank	Maintainability
100 - 20	A	Very high
19 - 10	B	Medium
9 - 0	C	Extremely low

At the bottom, there is a link: <https://radon.readthedocs.io/en/latest/>

*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.



#### 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:

Name	Date modified	Type	Size
.pytest_cache	12/02/2022 19:07	File folder	
.vscode	12/02/2022 18:54	File folder	
__pycache__	12/02/2022 19:21	File folder	
server.py	12/02/2022 19:13	Python File	1 KB
test_server.py	12/02/2022 19:21	Python File	1 KB

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "server.py - server -". The left sidebar has icons for file operations, search, and other development tools. The main editor area displays the following Python code:

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def digitpol():
7     return "Welcome to Digitpol"
8
9 if __name__ == '__main__':
10    app.run(debug=True)
```

Figure 32: Simple flask app

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "server.py 1" and "test\_server.py X". The main editor area displays the following Python code:

```
1 # test_server.py
2 from server import app
3
4 def test_server():
5     response = app.test_client().get('/')
6
7     assert response.status_code == 200
8     assert response.data == 'Welcome to Digitpol'
```

A tooltip window titled "Assertions in PyTest" provides information about Pytest assertions:

Pytest assertions are checks that return either True or False status. In Python Pytest, if an assertion fails in a test method, then that method execution is stopped there. The remaining code in that test method is not executed, and Pytest assertions will continue with the next test method.

<https://www.guru99.com/pytest-tutorial.html>

Figure 33: pytest codes

```

C:\ Command Prompt
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Neelam Pirbhai-Jetha>cd C:\Users\Neelam Pirbhai-Jetha\Desktop\server

C:\Users\Neelam Pirbhai-Jetha\Desktop\server>py.test
===== test session starts =====
platform win32 -- Python 3.9.5, pytest-7.0.0, pluggy-1.0.0
rootdir: C:\Users\Neelam Pirbhai-Jetha\Desktop\server
collected 1 item

test_server.py F [100%]

===== FAILURES =====
test_server

def test_server():
    response = app.test_client().get('/')

    assert response.status_code == 200
>       assert response.data == 'Welcome to Digitpol'
E           AssertionError: assert b'Welcome to Digitpol' == 'Welcome to Digitpol'
E             + where b'Welcome to Digitpol' = <WrapperTestResponse 19 bytes [200 OK]>.data

test_server.py:8: AssertionError
===== short test summary info =====
FAILED test_server.py::test_server - AssertionError: assert b'Welcome to Digitpol' == 'Welcome to Digitpol'
===== 1 failed in 0.47s =====

C:\Users\Neelam Pirbhai-Jetha\Desktop\server>

```

*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=PLosiE80TeTs4UjLw5MM6OjgkjFeUxCYH> + check: [https://github.com/CoreyMSchafer/code\\_snippets/tree/master/Python/Flask\\_Blog](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=zRwy8qtgJ1A>

#### **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.dini.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=OcD52IXq0e8> [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](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](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](https://www.tutorialspoint.com/cryptography_with_python/index.htm) [Accessed 12 February 2022]