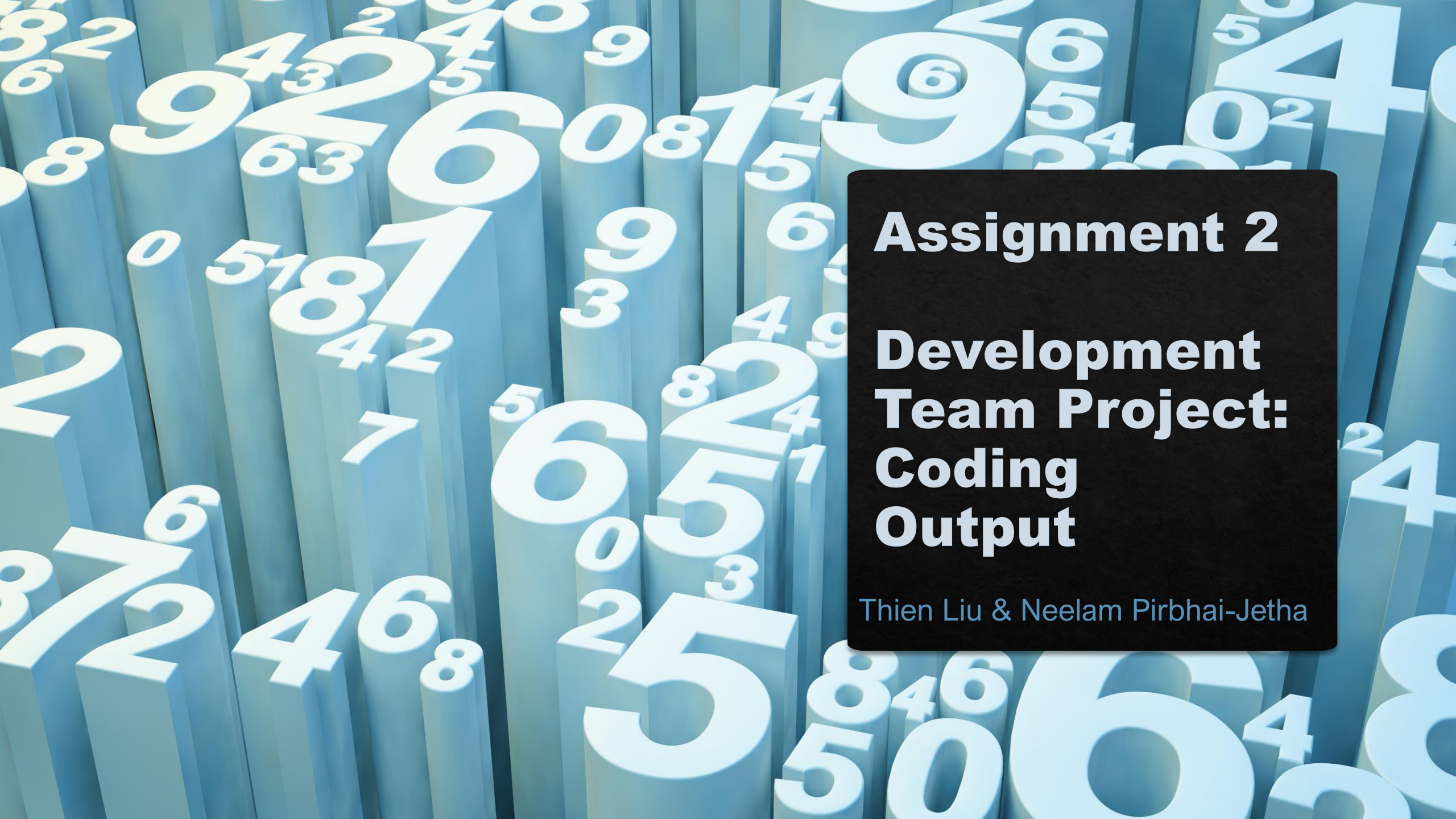


Assignment 2

Development Team Project: Coding Output

Thien Liu & Neelam Pirbhai-Jetha

Presentation Outline

Introduction &
Objective of this app

Demo

Security Features

Testing

1. INTRODUCTION

USERS:

Main Admin

Create accounts+Full CRUD operations

Internal Users

Full CRUD operations of their documents+Read

External Users

Read Only

- 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 system 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).

flask-authorize.readthedocs.io/en/latest/

Flask Authorize

About

Flask-Authorize provides utilities to help with user/role/group based app permissions and content CRUD authorization.

Useful Links

Flask-Authorize @ PyPI
Flask-Authorize @ github
Issue Tracker

Table of Contents

Overview

Flask-Authorize is a Flask extension designed to simplify the process of incorporating Access Control Lists (ACLs) and Role-Based Access Control (RBAC) into applications housing sensitive data, allowing developers to focus on the actual code for their application instead of logic for enforcing permissions. It uses a unix-like permissions scheme for enforcing access permissions on existing content, and also provides mechanisms for globally enforcing permissions throughout an application.

There are quite a few packages designed to simplify the process of adding ACLs and RBAC to a Flask application:

- Flask-Principal
- Flask-ACL
- Flask-RBAC
- Flask-Security

And each provides a different developer experience and makes different assumptions in their design. This package is yet another take at solving the same problem, resulting in a slightly different development experience when working with Flask applications. The developers of this package recommend you check out these alternatives along with Flask-Authorize to see if they fit your needs better.

https://flask-authorize.readthedocs.io/en/latest/

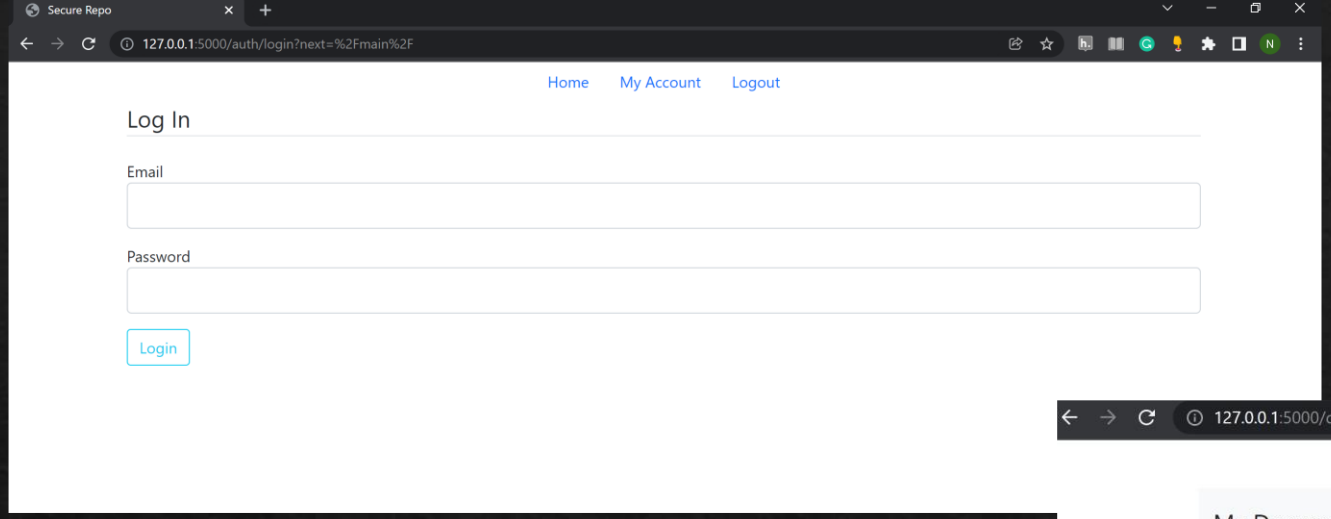
Hierarchical authentication: flask-authorize, a library in flask to give different roles to users

2. QUICK DEMO

Running flask on the command prompt

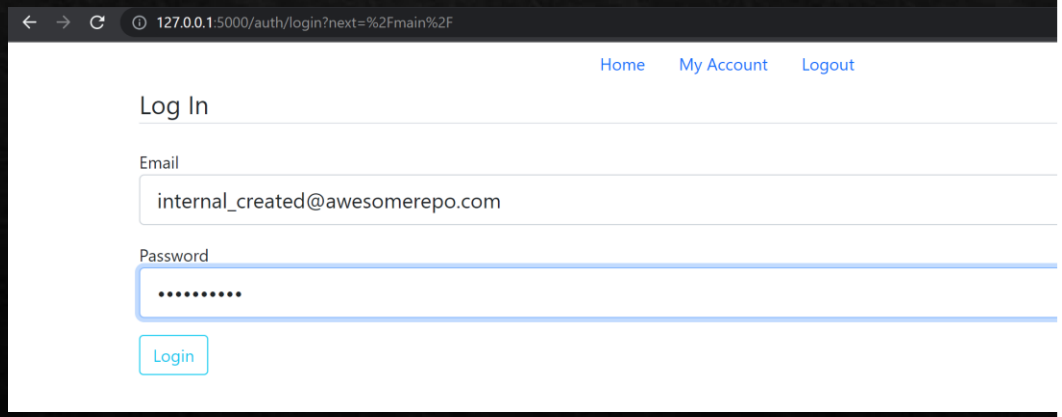
- cd C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop
- flask run

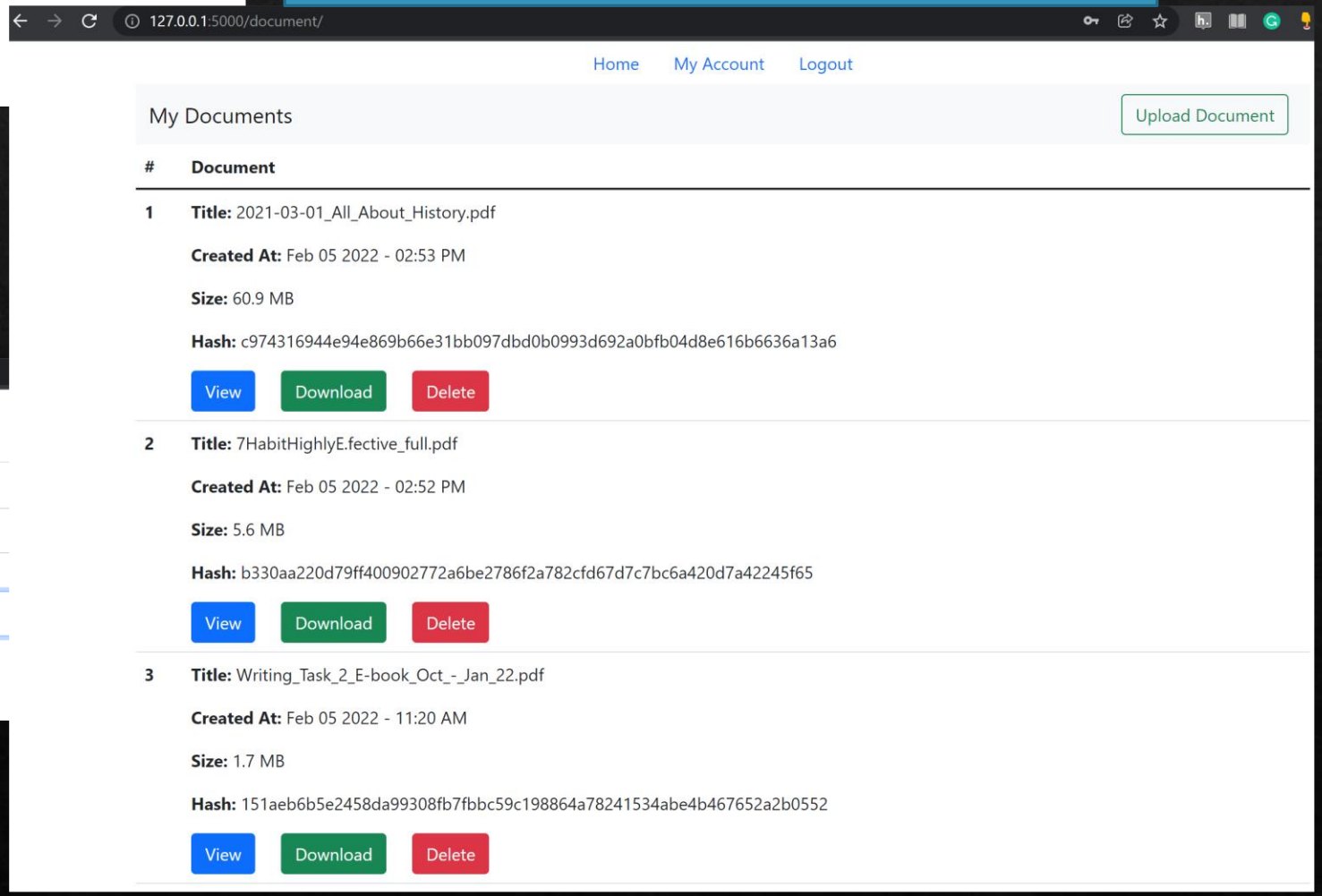
```
C:\Users\Neelam Pirbhai-Jetha>cd C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop

C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\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)
```

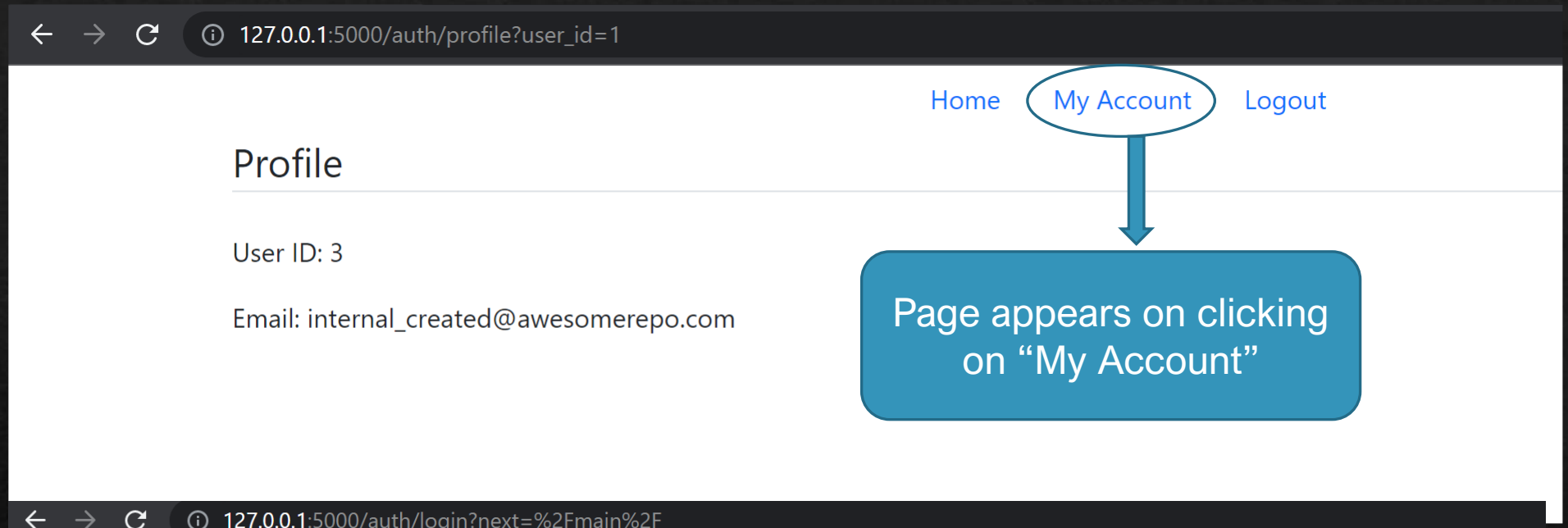


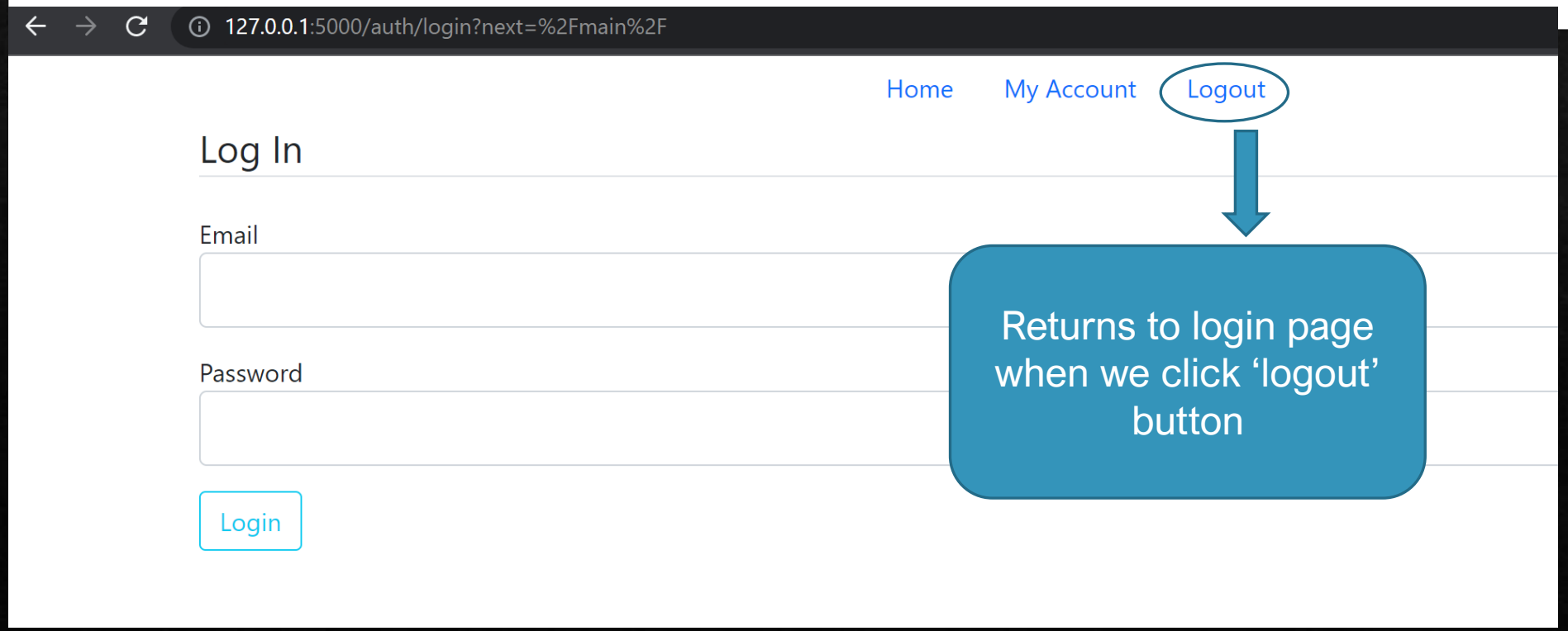
Go to create_user.py to get the login & password

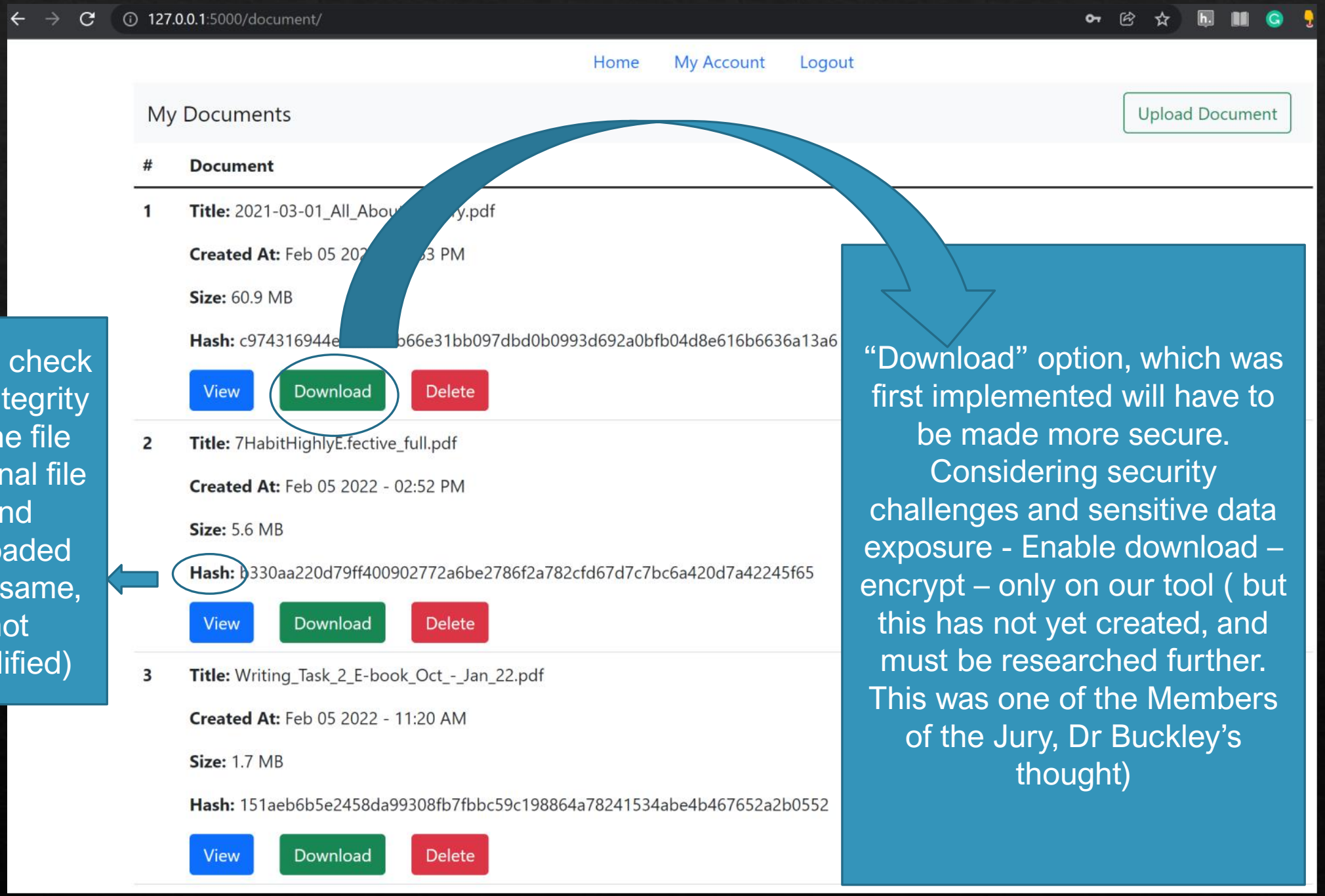
internal_created@awesomerepo.com
Qwerty@123






127.0.0.1:5000/auth/profile?user_id=1
Home
My Account
Logout
Profile
User ID: 3
Email: internal_created@awesomerepo.com
Page appears on clicking on “My Account”


127.0.0.1:5000/auth/login?next=%2Fmain%2F
Home
My Account
Logout
Log In
Email
Password
Login
Returns to login page when we click ‘logout’ button



Hash: check the integrity of the file (original file and uploaded file= same, not modified)

"Download" option, which was first implemented will have to be made more secure. Considering security challenges and sensitive data exposure - Enable download – encrypt – only on our tool (but this has not yet created, and must be researched further. This was one of the Members of the Jury, Dr Buckley's thought)

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)

3. SECURITY

Time constraints: other securities – adding fernet/cryptography

Security features of our app

- Encryption - bcrypt
- Authentication – authorisation (validators)
- manage session timeout

bcrypt

create_user.py

```python
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)
```

session.permanent = True
(sets after user successfully log in)
However, Dr Buckley, Member of the Jury suggested to reduce the session timeout (240 mins, being too long for the session)

config.py

```python
class Testing
    SECRET_KEY = os.                KEY_TEST', os.urandom(24))
    DATABASE_NAME = "awesome_repo_test.db"
    SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, DATABASE_NAME)
    ADMIND_DASHBOARD_VISIBLE = True
class ProductionConfig(Config):
    SECRET_KEY = os.getenv('SECRET_KEY_PROD', os.urandom(24))
    DATABASE_NAME = "awesome_repo_prod.db"
    SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, DATABASE_NAME)
    ADMIND_DASHBOARD_VISIBLE = False

config = {
```

config.py > 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      ADMIND_DASHBOARD_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      ADMIND_DASHBOARD_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      ADMIND_DASHBOARD_VISIBLE = False
30
31  config = {
32      'development': DevelopmentConfig,
33      'testing': TestingConfig,
34      'production': ProductionConfig
35  }
```

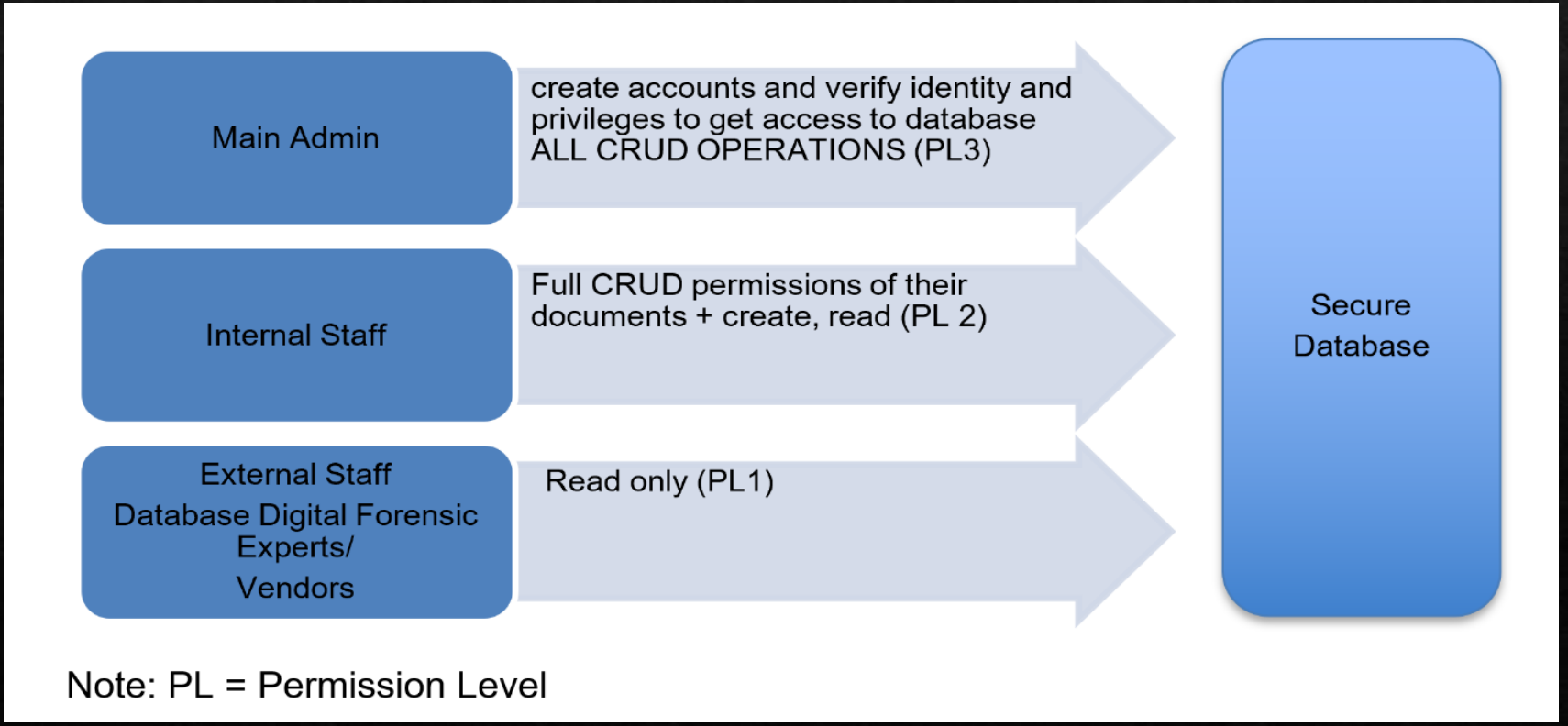
Three different environmental configurations

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

Modifications in the development config won't affect the others.

New features or issue fix – can be done in the development config, and then deployed to the testing config environment to verify if they pass the testing, then go public for the user (production config-deployment)

- Users do not register
- Admin creates and activates account
- Admin has full control of the system.
- Internal and external – have different permissions, but this has not been done yet



REGEX (Regular Expressions)

```
"""
There are various Python
packages and APIs
available to use instead
of Regex:
email-validator
pylsEmail
py3-validate-email
"""
```

We used wtforms.validators

```
regex.py > ...
"""
validate email addresses in Python, using Regular Expressions
Regular Expressions, often shortened as regex,
- are a sequence of characters used to check whether a pattern exists in a given text (string) o
- are, for instance, used at the server side to validate the format of email addresses or passwo
registration
"""
#The re module contains classes and methods to represent and work with Regular Expressions in Py
import re

"""The re.compile() method compiles a regex pattern into a regex object
regex = re.compile(r'([A-Za-z0-9]+[.-_])*[A-Za-z0-9]+@[A-Za-z0-9-]+(\.[A-Z|a-z]{2,})+'))
but the above codeline not working"""

regex='^[a-z0-9]+[\._]?[\a-z0-9]+[@]\w+[. ]\w{2,3}$'

"""
explaining the above regular expressions: (refer to Unit 4 - class notes)
^ A caret: Matches a pattern at the start of the string
$ Dollar sign. Matches the end of the string.
? Question mark: Checks if the preceding character appears exactly zero or one time.
And Specifies a non-greedy version of +, *
w Lowercase w: Matches any single letter, digit, or underscore.
\ backslash: If the character following the backslash is a recognized escape character, then the
· Else the backslash () is treated like any other character and passed through.
· It can be used in front of all the metacharacters to remove their special meaning
"""

"""
define a function to validate email
pass the regular expressionand the string into the fullmatch() method
"""
def isValid(email):
    if re.fullmatch(regex, email):
        print("Valid email")
    else:
        print("Invalid email")

# Driver Code
if __name__ == '__main__':

    # Enter the email
    email = "goodmorning1@gmail.com"
    # calling run function
    isValid(email)
```

```python
# Driver Code
if __name__ == '__main__':

    # Enter the email
    email = "goodmorning1@gmail.com"
    # calling run function
    isValid(email)

    email = "my.email@hotmail-go.org"
    isValid(email)

    email = "goodmorning622022.com"
    isValid(email)



"""
Sources:
GeeksforGeeks (August 8, 2021) "Check if email address valid or not in Python". Available from:
Gupta, R. (February 7, 2021) "How To Validate An Email Address In Python". Available from: https
Jaiswal, S. (2020) Python Regular Expression Tutorial. Available from: https://www.datacamp.com/
Peoples, C. (2021), Lecture Notes, Secure Software Development SSDCS_PCOM7E, University of Essex

```

Some e-mail addresses that are being checked

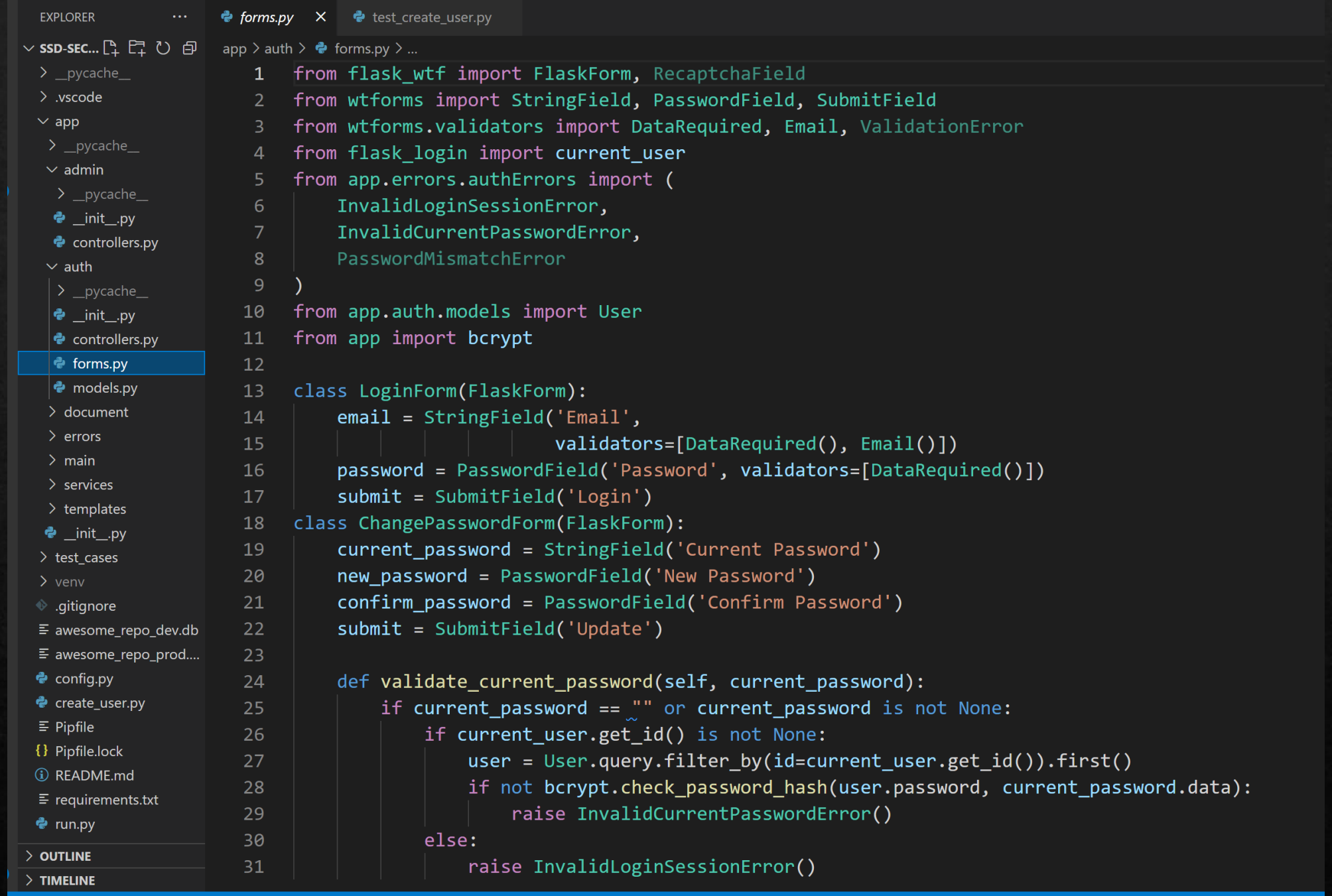
PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE

Python Debug Console

```
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional>  c:; cd 'c:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional
 & 'C:\Users\Neelam Pirbhai-Jetha\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Neelam Pirbhai-Jetha\.vscode\extensions\ms-python.p
hon-2022.0.1786462952\pythonFiles\lib\python\debugpy\launcher' '59616' '--' 'c:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional\re
x.py'
Valid email
Invalid email
Invalid email
```

Valid or not?

Screenshot of email validator of our app (instead of Regex)



config.py

config.py > TestingConfig

```python
import os
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 = os.urandom(24)

# Session
PERMANENT_SESSION_LIFETIME = timedelta(minutes=240)

class DevelopmentConfig(Config):
    SECRET_KEY = os.getenv('SECRET_KEY_DEV', os.urandom(24))
    DATABASE_NAME = "awesome_repo_dev.db"
    SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, DATABASE_NAME)
    ADMIND_DASHBOARD_VISIBLE = True
class TestingConfig(Config):
    SECRET_KEY = os.getenv('SECRET_KEY_TEST', os.urandom(24))
    DATABASE_NAME = "awesome_repo_test.db"
    SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, DATABASE_NAME)
    ADMIND_DASHBOARD_VISIBLE = True
class ProductionConfig(Config):
    SECRET_KEY = os.getenv('SECRET_KEY_PROD', os.urandom(24))
    DATABASE_NAME = "awesome_repo_prod.db"
    SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, DATABASE_NAME)
    ADMIND_DASHBOARD_VISIBLE = False

config = {
```

GENERATING SECRET KEYS: random was used

not-secure-otp.py > ...

```python
# Python Program for simple OTP genertaor

"""
random is a module in Python that generates random numbers
However, being completely deterministic, it is not suitable for all purposes,
and is completely unsuitable for cryptographic purposes.

The pseudo-random generators of this module should not be used for security purposes.
For security or cryptographic uses, see the secrets module

"""
import random as r
# function for otp generation (otpgen)
def otpgen():
    otp=""
    for i in range(6):
        otp+=str(r.randint(1,9))
    print ("Your One Time Password is " + otp)

otpgen()

"""
Sources:
https://docs.python.org/3/library/random.html
"""
```

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE Python Debug

```
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional>  c:; cd 'c:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNME
 & 'C:\Users\Neelam Pirbhai-Jetha\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Neelam Pirbhai-Jetha\.vscode\ex
hon-2022.0.1786462952\pythonFiles\lib\python\debugpy\launcher' '49696' '--' 'c:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2
secure-otp.py'
Your One Time Password is 245463
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\ASSIGNMENT2\codes_additional>
```

Batterv statu

Random generator is not secure. Use ‘secrets’

+ OTP has not been included yet in the Flask app

filesError.py | test_create_user.py

app > errors > filesError.py > FileAlreadyExistsError

```python
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__()
```

app > errors > authErrors.py > InvalidLoginSessionError

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

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

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

Codes written to raise errors if input of data is wrong

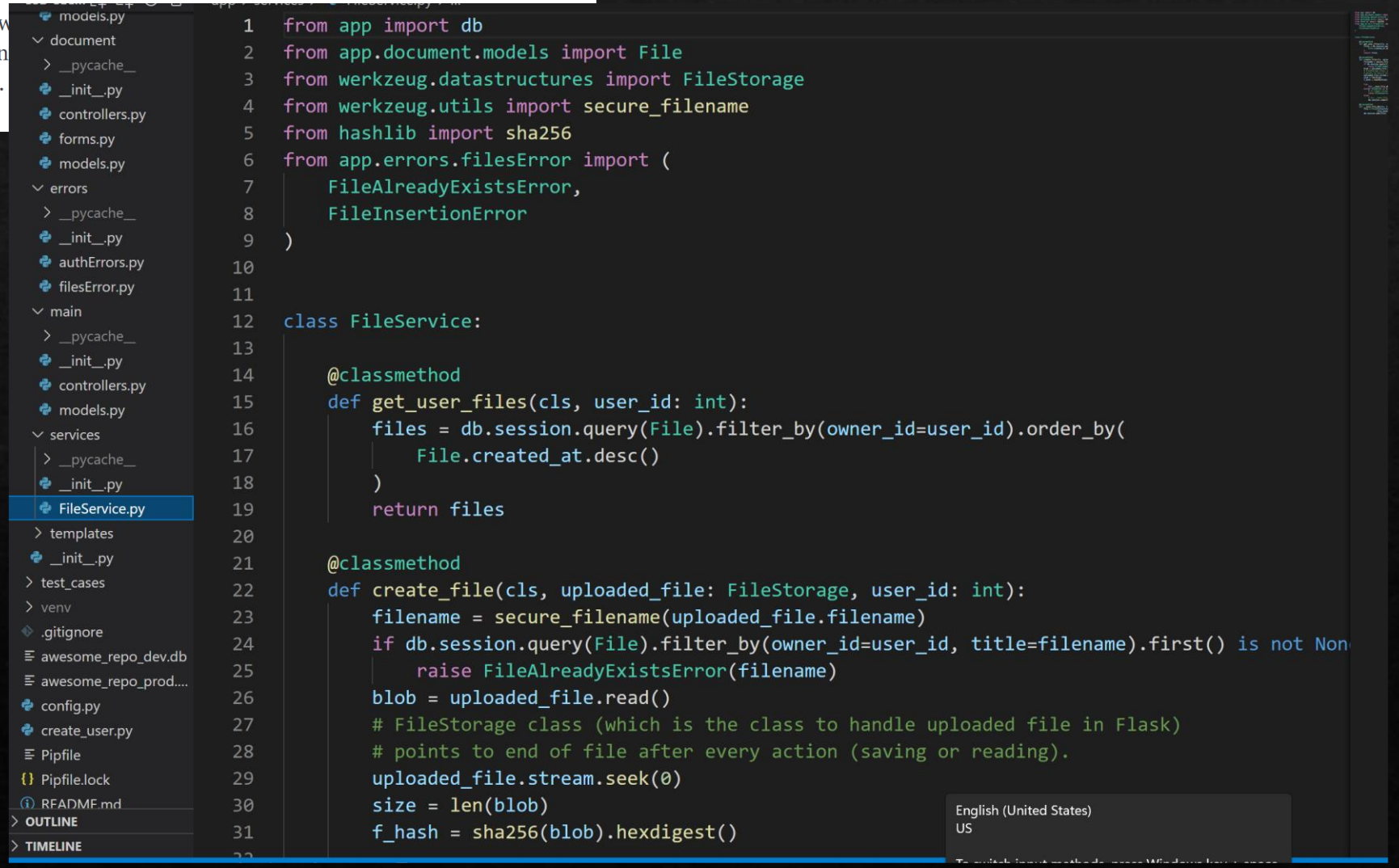
flask-hashing.readthedocs.io/en/latest/

Flask-Hashing

Flask-Hashing is a Flask extension that provides an easy way to hash data and check a hash of a value against a given hash. Flask-Hashing uses *hashlib* to actually hash data.

The main use case for hashing in w
an application may have a differen
choices are not password-specific.

Even if we use these libraries for our development phase, there might be some changes (in progress)



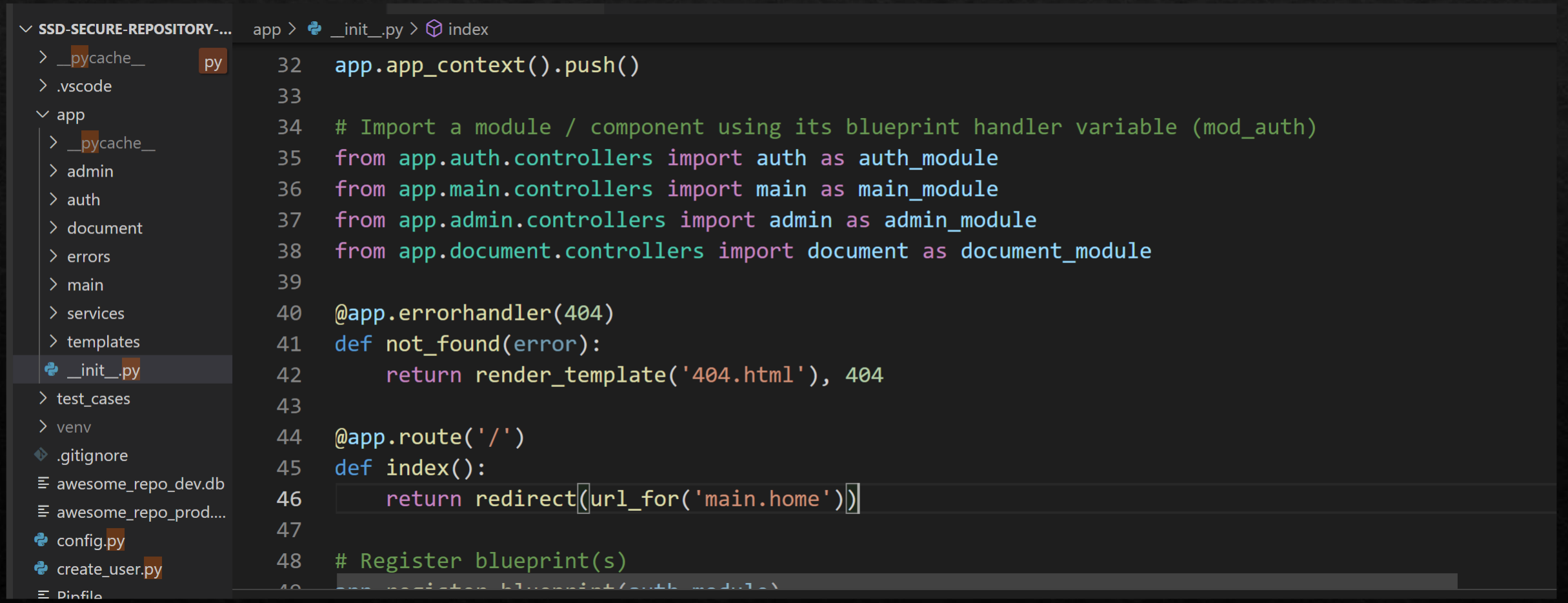
4. TESTING

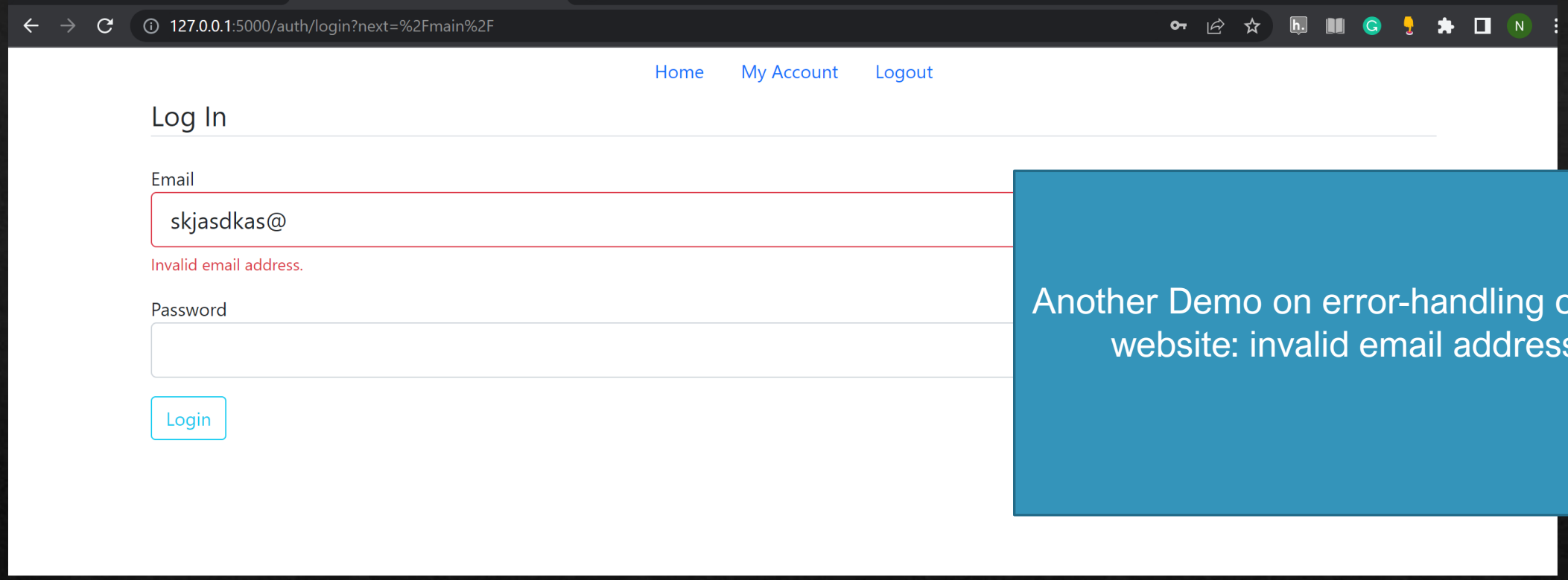
- The software testing activity is very expensive, critical and complex, and a software without testing is most dangerous and leads to more expensiveness. Software testing is, therefore, an essential activity before the release of any software (Madhavi, 2016; Syaikhuddin et al, 2018).

- We have tried to apply some of the tools we have learnt in this class (e.g pylint, flake8, mccabe, cyclomatic complexity)

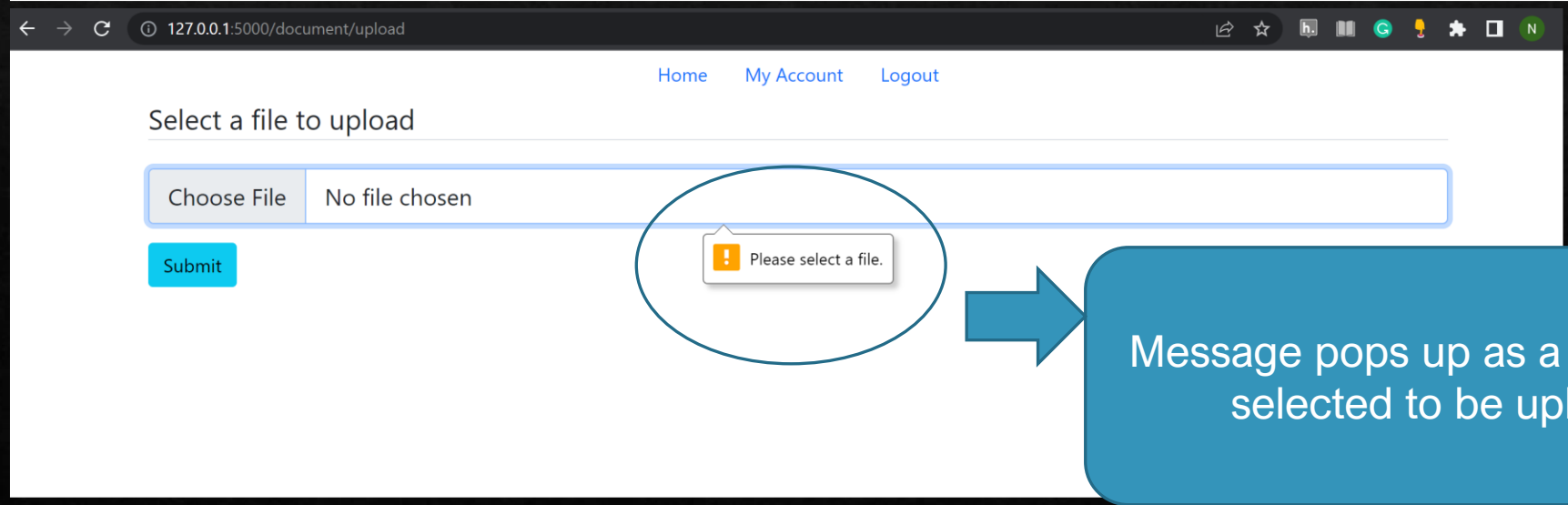
- error/exception handling: manually (wrong password, wrong email)

- This is part of the ERROR HANDLER: if our system cannot find the route, it will redirect to the 404.html template





Another Demo on error-handling on our website: invalid email address



Message pops up as a file must be selected to be uploaded

Testing: pylint

secure_otp.py > ...

```python
"""
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>
```

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

Pylint testing on create_user.py

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE powershell

```
PS C:\Users\Neelam Pirbhai-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:1:0: C0114: 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 'add' 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 Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>
```

Pylint testing on run.py

```
13    PERMANENT_SESSION_LIFETIME = timedelta(minutes=240)
```

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE

```
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> pylint run.py
************* Module run
run.py:3:0: C0304: Final newline missing (missing-final-newline)
run.py:1:0: C0114: Missing module docstring (missing-module-docstring)

-----------------------------------
Your code has been rated at 0.00/10

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

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE powershell +

```
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 config.py
config.py:5:1: E302 expected 2 blank lines, found 0
config.py:11:1: W293 blank line contains whitespace
config.py:13:1: E305 expected 2 blank lines after class or function definition, found 1
config.py:15:1: E302 expected 2 blank lines, found 1
config.py:18:80: E501 line too long (82 > 79 characters)
config.py:20:1: E302 expected 2 blank lines, found 0
config.py:23:80: E501 line too long (82 > 79 characters)
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: E305 expected 2 blank lines after class or function definition, found 1
config.py:35:2: W292 no newline at end of file
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>
```

Testing flake8

"Flake8 = Python library to check the code base against coding style (PEP8), programming errors (like "library imported but unused" and "Undefined name") and to check cyclomatic complexity" [Freitas, V. "How to use Flake8". https://simpleisbetterthancomplex.com/packages/2016/08/05/flake8.html]

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE pow

```
PS C:\Users\Neelam Pirbhai-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:10:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:11:14: 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:12:19: 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:14:15: 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:18:16: E251 unexpected spaces around keyword / parameter equals
create_user.py:18:42: W291 trailing whitespace
create_user.py:19:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:19:19: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:18: W291 trailing whitespace
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:24:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:25:14: E251 unexpected spaces around keyword / parameter equals
create_user.py:25:16: E251 unexpected spaces around keyword / parameter equals
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:27:13: 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:28:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:28:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:35:1: E305 expected 2 blank lines after class or function definition, found 1
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>
```

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

PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE powershell

```
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 --max-complexity 10 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:10:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:11:14: 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:12:19: 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:14:15: 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:18:16: E251 unexpected spaces around keyword / parameter equals
create_user.py:18:42: W291 trailing whitespace
create_user.py:19:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:19:19: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:13: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:20:18: W291 trailing whitespace
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:24:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:25:14: E251 unexpected spaces around keyword / parameter equals
create_user.py:25:16: E251 unexpected spaces around keyword / parameter equals
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:27:13: 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:28:15: E251 unexpected spaces around keyword / parameter equals
create_user.py:28:17: E251 unexpected spaces around keyword / parameter equals
create_user.py:35:1: E305 expected 2 blank lines after class or function definition, found 1
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>
```

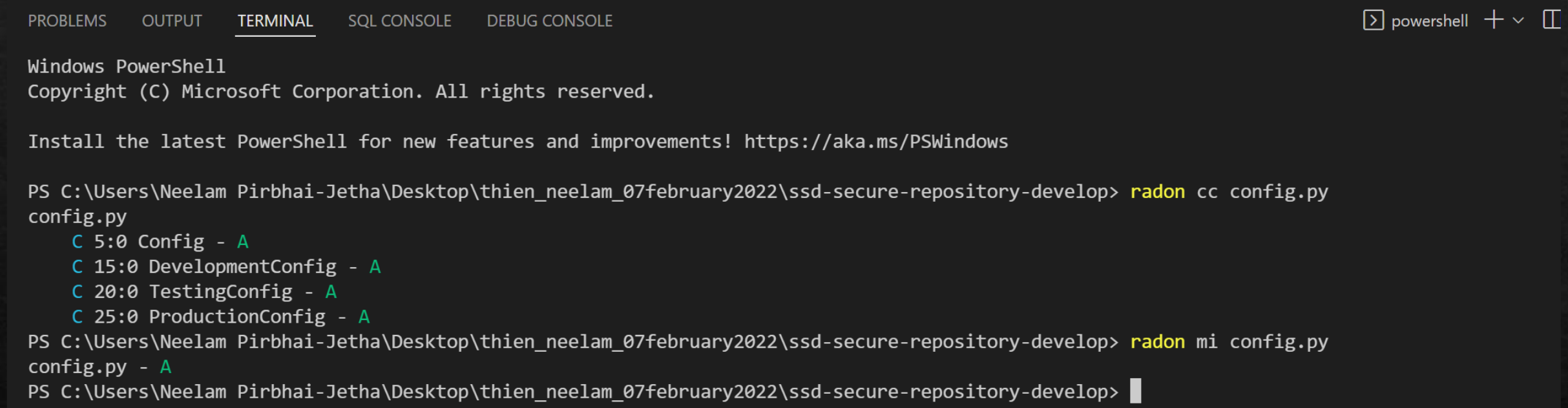
PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE powershell

```
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop> flake8 --max-complexity 10 config.py
config.py:5:1: E302 expected 2 blank lines, found 0
config.py:11:1: W293 blank line contains whitespace
config.py:13:1: E305 expected 2 blank lines after class or function definition, found 1
config.py:15:1: E302 expected 2 blank lines, found 1
config.py:18:80: E501 line too long (82 > 79 characters)
config.py:20:1: E302 expected 2 blank lines, found 0
config.py:23:80: E501 line too long (82 > 79 characters)
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: E305 expected 2 blank lines after class or function definition, found 1
config.py:35:2: W292 no newline at end of file
PS C:\Users\Neelam Pirbhai-Jetha\Desktop\thien_neelam_07february2022\ssd-secure-repository-develop>
```

bit ⊗ 0 ⚠ 0 Connect Server not selected Ln 23, Col 83 Spaces: 4 UTF-8 LF Python

Cyclomatic complexity

- Radon is a Python tool that computes various metrics from the source code. Radon can compute:
- **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)
- Source: https://pypi.org/project/radon/
- In cmd: pip install radon



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

The **cc** command

This command analyzes Python source files and compute Cyclomatic Complexity. The output can be filtered by specifying the `-n` and `-x` flags. By default, the complexity score is not displayed, the option `-s` (show complexity) toggles this behaviour. File or directories exclusion is supported through glob patterns. Every positional argument is interpreted as a path. The program then walks through its children and analyzes Python files. Every block will be ranked from A (best complexity score) to F (worst one). Ranks corresponds to complexity scores as follows:

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 **mi** command

This command analyzes Python source code files and compute the Maintainability Index score. Every positional argument is treated as a starting point from which to walk looking for Python files (as in the **cc** command). Paths can be excluded with the `-e` option. The Maintainability Index is always in the range 0-100. MI is ranked as follows:

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

Options

https://radon.readthedocs.io/en/latest/

Pip install pytest

- **PyTest** is a testing framework that allows users to write test codes using Python programming language. It helps to write simple and scalable test cases for databases, APIs, or UI. PyTest is mainly used for writing tests for APIs. It helps to write tests from simple unit tests to complex functional tests.
- **Which is better – pytest or unittest?**
  Although both the frameworks are great for performing testing in python, pytest is easier to work with. The code in pytest is simple, compact, and efficient.
  For unittest, we will have to import modules, create a class and define the testing functions within that class. [https://www.pythonpool.com/python-unittest-vs-pytest/#:~:text=Which%20is%20better%20%E2%80%93%20pytest%20or,testing%20functions%20within%20that%20class.]

- Create test_filename.py
- In the Terminal,
- py.test filename.py
- Check https://www.youtube.com/watch?v=OcD52lXq0e8

Equivalence Partitioning Testing or Equivalence Class Partitioning = black box testing

- To test, for instance, if the right number of digits have been entered (for the OTP).
- Find the code for equivalence Testing

Summing up

- We are thinking about
  - a dashboard of threat monitoring, a system audit logs to track the user’s activities by looking at the logs (detect unusual activities)
  - Cryptography (fernet,…)