National College of Ireland

<BSHCYB4>

<Cybersecuirty>

<Academic Year i.e. 2022/2023>

<Conor Keller>

<x18353923>

< x18353923@student.ncirl.ie>

<SafePass>

Technical Report

Contents

Executive Summary

1.0 Introduction

1.1. Background

A web-based password manager application was the most popular project created by cyber security students last year. Knowing myself and my capabilities, I chose to create my own version of the same basic idea because my main goal for this module is to create an application I can call my own and be proud of without putting too much pressure on myself by coming up with an original idea that could result in a lot of work.

1.2. Aims

Through my research on this subject, I have come across a wide variety of "Password Safes" on the internet. However, most of these applications seem to be lacking a feature that would prevent malleolus hackers from accessing the given users' Master Accounts because the main users used a password that was likely to be used across the board. In my application, I aim to address this issue.

1.3. Technology

From my current research the following languages are what I intend to use, these languages are new to me so I will require a lot of practice and understanding before I can develop the application itself but as follows these are the languages, I will use to create my application. HTML CSS, Java, Django framework & Python, I have some familiarity with these Languages having used them in previous college projects but to use them all together in conjunction to create my application will be anew task in itself for me.

My main text for this application will be Microsoft visual studio as I have been using this editor for the past two years, so I feel comfortable and confidence using this in the creation of the application, plus the benefits of using the above-mentioned languages can be used within the VsCode as it provides a simplicity and a smooth experience.

I will be using SQL lite as my database for this application, from my current research it seems to be a popular choice for this type of application, I am not complete familiar with this type of DB so I will have to do more research and watch more tutorials on how to proceed with using this in conjunction with my chosen languages.

From my current research the following algorithms that are required to complete this application are as follows, just a side note I am not familiar with these at all as I have not needed to use any form of algorithms in previous projects so this will be a taxing process on myself, but the result will showcase a sound project. SHA256 - Predictable password hash, Argon2 - Unpredictable password hash for the database, pbkdf2 - Generate the vault key &AES256 - Encrypt and decrypt the vault.

Note: Current technologies used, will need to re-evaluate project proposal.

- I'm using Microsoft SQL Management Studio to link my user database to a local database where I'm storing user data.
- I created my project using the Django core framework, which facilitates the development of server-side web applications.

1.4. Structure

2.0 System

2.1. Requirements

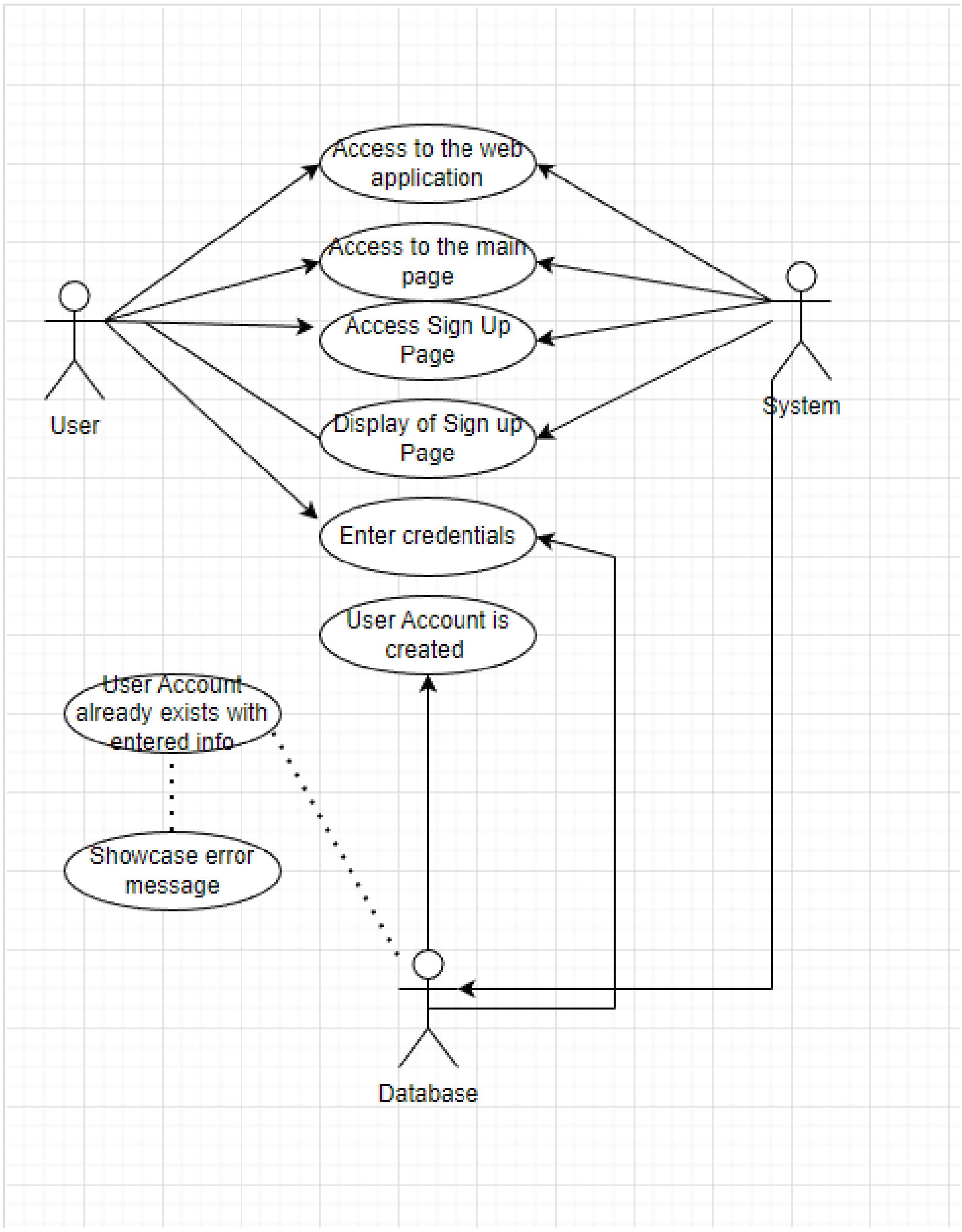
Current Requirements for the project:

- Allows users a means of creating an account for the application
- Offer 2 step verification for account creation (Email)
- Offer 2 step verification for account log in (Email)
- Allow users to create “Vaults” that acts as Cards on the web aplication that contain information which contains email & Passwords for sites of their choosing.
- Allow created “Vaults” to be deleted and updated.
- Allow users to generate Password reset key via mail
- Allow users to copy Email and Passwords from vault to their system keyboard
- Allow Users to logout of the application

2.1.1. Functional Requirements

The functional needs for this project are listed in descending order of importance, with the most critical criteria listed first. User integrity, application functionality, security, and secrecy are given a ranking depending on their importance.

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1

User Sign Up

2.1.1.3. Description & Priority

Scope: The given scope will be the registration of the user

Description:

This prerequisite describes how to sign up using their email and a chosen password, If the user doesn't already have an account, they can establish one with their email address. This is necessary since accessing any of the application's features requires the user to be logged in, which is the most important requirement.

Flow Description:

This Use Case showcases the process of user Sign Up

Precondition:

The User should have a valid email address & Password requirement must include uppercase/lowercase lettering followed by symbols & numbers

Activation:

The application is activated upon running the SQL database, running the project in visual studio, and once done the application will produce the webpage.

Main flow:

- Django runs SQL database in the background
- User builds project in visual studio
- User know has access to the web page
- Django displays the Main page
- User may login
- Credentials are inputted
- Identity verifies information
- Django showcases the application main home page

Alternate flow:

A1: Valid account already exists
- User navigates themselves to the sign up page
- Django showcases registration page
- User inputs credential information
- Django detects invalid credential
- The user is guided back to point 3 of the main flow

Exceptional flow

E1: Loss of internet connectivity
- Internet access is lost
- Error message is displayed
- Connection is regained and the user is guided back to point one of the main flow

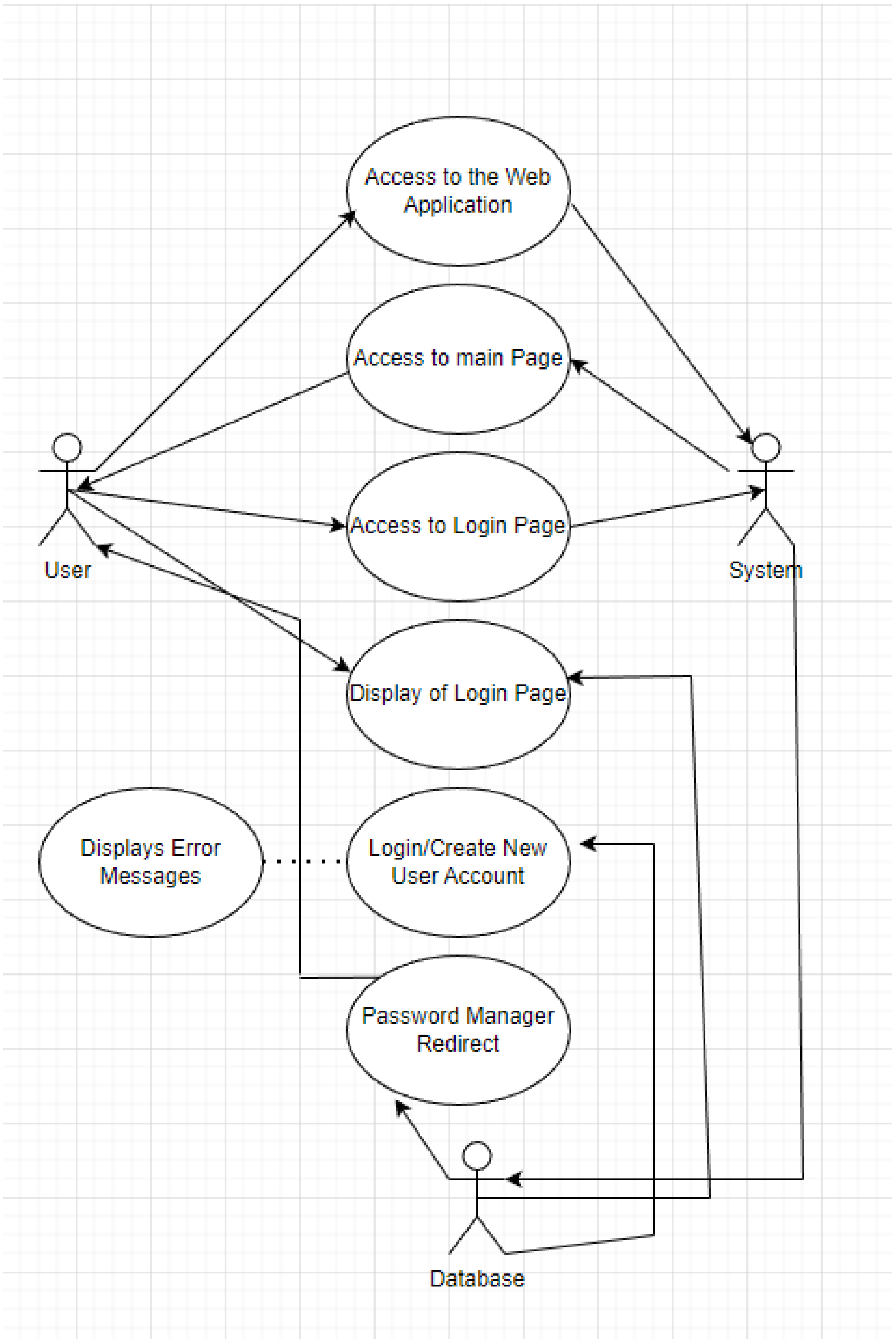
Termination

- The use case is terminated once a successful login has been achieved

Post condition

- The database is monitoring actions proceeded by the user

2.1.1.4. Use Case Diagram



2.1.1.5. Requirement 2

User Login

2.1.1.6. Description & Priority

Scope: The given scope will be the Login of the user

Description:

This prerequisite describes how to log in using their email and chosen password, If the user doesn't already have an account, they can establish one with their email address. This is necessary since accessing any of the application's features requires the user to be logged in, which is the most important requirement.

Flow Description: This Use Case showcases the process of Log In

Precondition:

The User should have a valid email address & Password requirement must include uppercase/lowercase lettering followed by symbols & numbers

Activation:

The application is activated upon running the SQL database, running the project in visual studio, and once done the application will produce the webpage.

Main flow:

- Django runs SQL database
- User builds project in visual studio
- User know has access to the web page
- Django displays the Main page
- User may login
- Credentials are inputted
- Identity verifies information
- Django showcases the application main home page

Alternate flow:

A1: Valid account does not exist
- User navigates themselves to the login page
- Django showcases Login page
- User inputs credential information
- Django detects invalid credential
- The user is guided back to point 3 of the main flow

Exceptional flow

E1: Loss of internet connectivity
- Internet access is lost
- Error message is displayed
- Connection is regained and the user is guided back to point one of the main flow

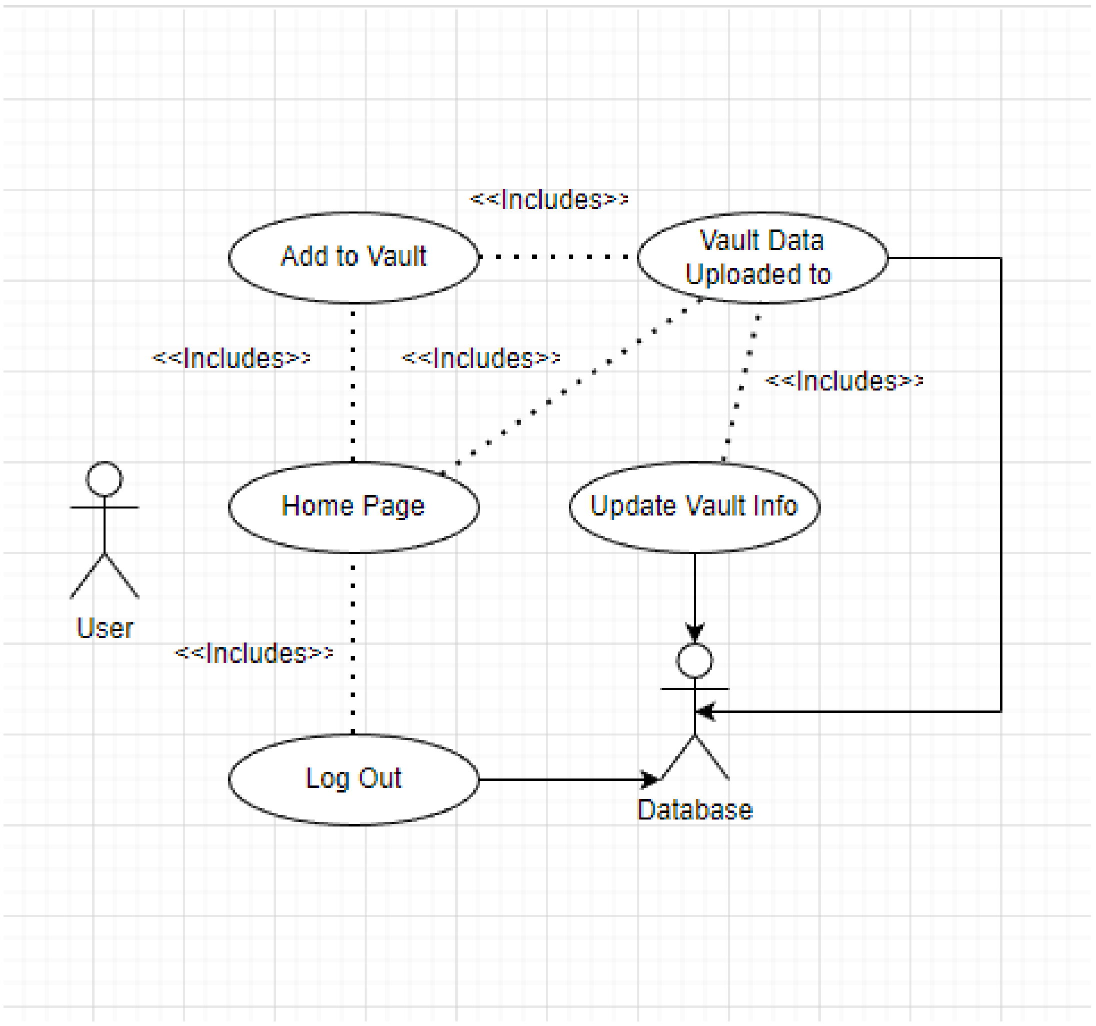
Termination

- The use case is terminated once a successful login has been achieved

Post condition

- The database is monitoring actions proceeded by the user

2.1.1.7. Use Case Diagram



2.1.1.8. Requirement 3

Password Vault Manager

2.1.1.9. Description & Priority

Scope:

The purpose of this use case is to introduce a user to the password manager's features and demonstrate how to utilize it. Additionally, it describes how the manager saves their login information to the database.

Description:

This requirement will be outlining the process for adding vaults to the application from the given users account once they’re logged in and what functionality is present once logged in such as adding new vaults, deleting/updating the information present, it is up to the most importance that the user is logged in with the correct credentials as without them the user will not be able to access their password vaults

Flow Description:

This use case details the interaction between the user and the password manager, database, and its functions. The use case also shows how the database is populated with data by the password manager.

Precondition:

The User must have the application running and must be logged in

Activation:

The application is activated upon running the application by doing the following c:\SafePass\AutoDjango\MAIN then python manage.py runserver

Main flow:

- User Access the main page
- User can add a new vault to their account
- User updates their account vaults by ether deleting/ updating their infomation
- System verifiers these changes and uploads it to the database

Alternate flow:

A1: Valid account does not exist

- User inputs credential information
- Incorrect User details entered
- Application displays error message for user
- The user is guided back to point 3 of the main flow

Exceptional flow

E1: Loss of internet connectivity

- Internet access is lost
- Error message is displayed
- Connection is regained and the user is guided back to point one of the main flow

Termination

- The use case is terminated once a successful login has been achieved and that the user can successfully added new vaults/ updated and deleted.

Post condition

- The database is monitoring actions proceeded by the user

2.1.2. Data Requirements

The strictest industry standards for data confidentiality, integrity, and availability should be met by the password manager in addition to giving total control over passwords. AES 256-bit encryption, end-to-end encryption, multi-factor authentication, and PBKDF2 are important security features to look for and will be researched accordingly.

2.1.3. User Requirements

The user should be able to generate a strong and memorable master password. To secure the password manager, you must select a long, complex password. However, if the phrase is just important to you, choose one that can withstand dictionary attacks. Allow multifactor authentication through SMS text / Email Confirmation. This protects users in the event of an attack when the master password has been compromised.Security, Confidentiality, Integrity, Usability, Consistency, and Reliability are all qualities that the application should offer its user base.

2.1.4. Environmental Requirements

This program doesn't have any audio capabilities, therefore external noises or interruptions won't affect the user experience. Since I believe that is the only environmental concern that may occur from a project with such a scope

2.1.5. Usability Requirements

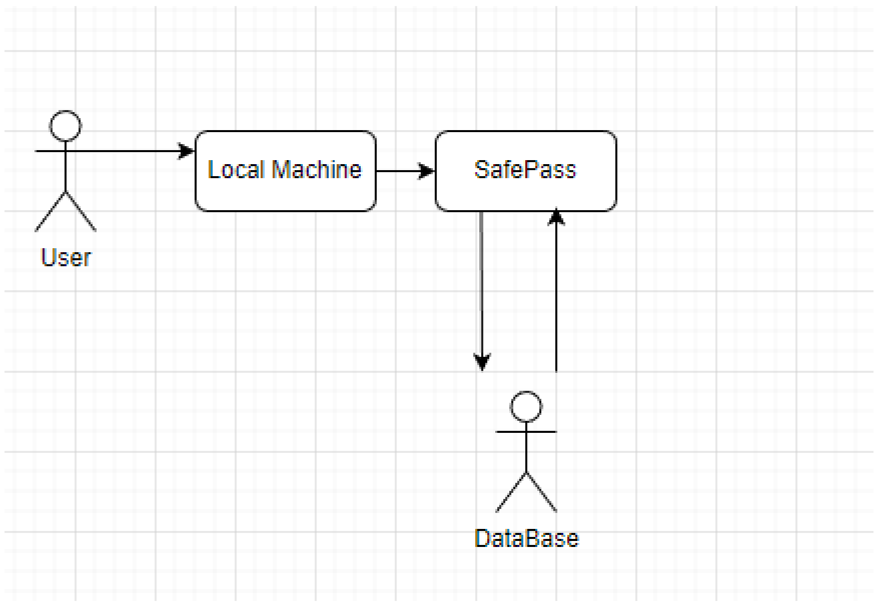
The application interface is clean and simple while also providing next to none on clutter for when the user accesses the application and within the application the users vaults. In any instance no user should require a guide or manual to use the password manager in its entirety. Error messages pop up in the middle of the screen when errors do occur which is eye catching and appealing

The usability requirements in the project include the following:

- **Efficiency of use:** Objectives may be attained fast and with little to no user mistake.
- **Intuitiveness:** The interface is straightforward to use and navigate; the functionality pages like signup/login/add vault, headers, footer and error messages are clear.
- **Low perceived workload:** Rather than being intimidating, hard, or annoying, the interface is simple to use.

2.2. Design & Architecture

In the applications completed form, it has been developed using HTML, CSS, JAVA, Python & Django within the framework of Visual Studio 2022



The systems Architecture as conceived in the above image; Any given user can use the application on their given local machine as long as that local machine has Python installed and updated to the most recent version. As SafePass is a Python-Django created application with multi-platform compatibility like the likes of mac os and linix which translate to mean that the application can be ran and used across platforms on any device such as a pc, laptop, mobile and tablet.

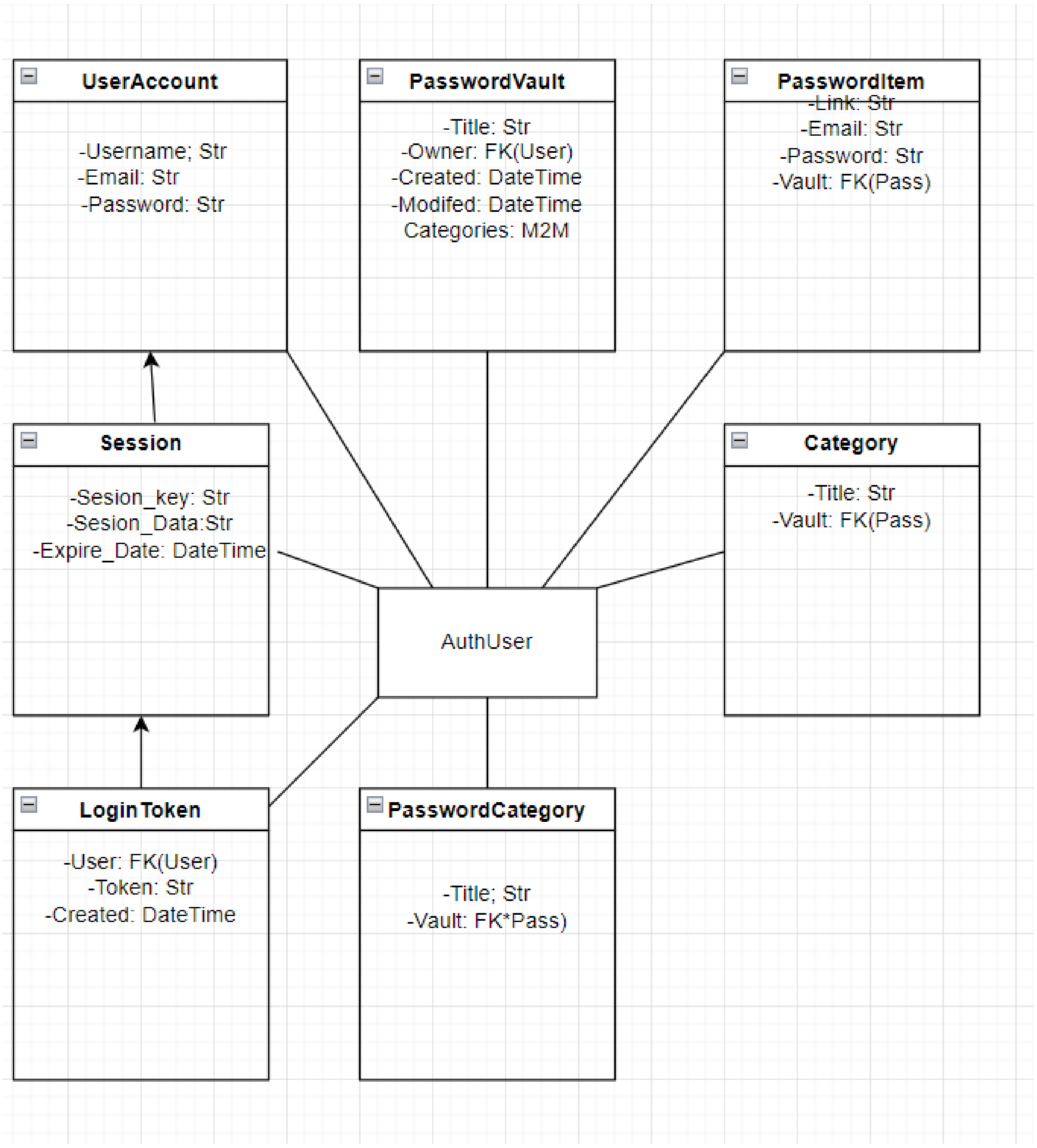
The application is functional in an live local environment as this is the only way to use the application, so in the instance the application is offline local then the functionality of the application will not be accessible.

Using Python and the Django framework, it was possible to create a powerful and secure password manager, The first step was to create a Django project and then define the models necessary for storing user accounts and passwords. The next step was to create the views and templates needed to display the password manager to the user base. Next was to ensure the security of the passwords, it is important to use encryption and hashing algorithms, in my case with Python thankfully it provides several libraries for this purpose, such as bcrypt or cryptography. In my case I used Cryptography, Additionally the Django framework includes built in features such as CSRF protection and user authentication, which further enhances the security of the password manager aka SafePass. With these tools, it was possible to create a robust and user-friendly password manager that can help users keep their sensitive information safe.

Python and Django were excellent candidates for developing SafePass because of their simplicity of use, versatility, and resilience. With these technologies, I was able to design a password management vault that is user-friendly, secure, and efficient, giving users peace of mind while managing many password

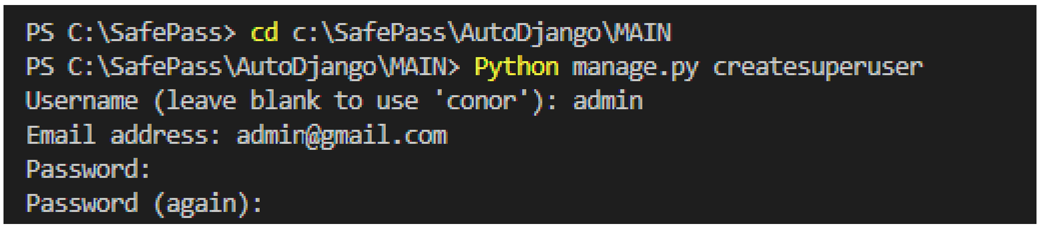
2.3. Implementation

Beta Use Case concept idea

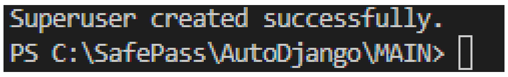


How I Run Application locally:

Creating an admin account

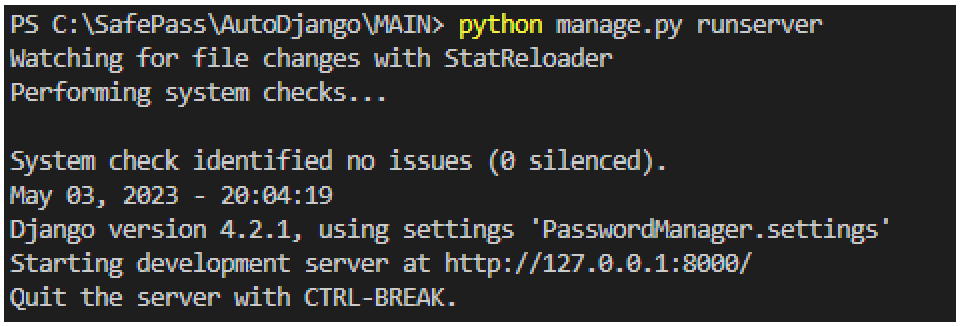


```
PS C:\SafePass> cd c:\SafePass\AutoDjango\MAIN
PS C:\SafePass\AutoDjango\MAIN> Python manage.py createsuperuser
Username (leave blank to use 'conor'): admin
Email address: admin@gmail.com
Password:
Password (again):
```



```
Superuser created successfully.
PS C:\SafePass\AutoDjango\MAIN> 
```

Runs server for application



```
PS C:\SafePass\AutoDjango\MAIN> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 03, 2023 - 20:04:19
Django version 4.2.1, using settings 'PasswordManager.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

This portion of the report details my progress to date as well as minor deviations from the initial implementation plan:

SafePass is a password manager vault application built with Django, Python, HTML, CSS, JavaScript, and SQLite.

Setting up the development environment was the first step in designing SafePass. I installed Python on my PC and then installed Django using pip, Python's package manager. Django is a high-level Python web framework that provides a set of tools and frameworks for rapidly and effectively constructing online applications. SQLite, a lightweight relational database management system, was also installed and used to store the user's login credentials.

I used the django-admin startproject command to establish a new Django project after configuring the development environment. This program established a new Django project with a basic directory structure and configuration files.

Then I used the python manage.py startapp command to establish a new Django app. This script creates a new app called PasswordManager within my Django project. Then named my application SafePass to supply the application with its own unquie ID

For designing the application I used HTML, CSS, and JavaScript to design the SafePass user interface. I made a basic HTML template with a header, a navigation bar, a content section, and a footer. CSS was utilized to style the various aspects of the user interface, such as the form fields, buttons, and navigation bar. I also utilized JavaScript to provide interaction to the user interface, such as showing success and problem messages when a password is saved or deleted.

I used cryptographic techniques such as encryption and hashing to ensure that users' login credentials are safely maintained. Cryptography is critical in safeguarding sensitive data, particularly when storing login passwords. I also used HTTPS to encrypt all communication between the user's browser and the application server.

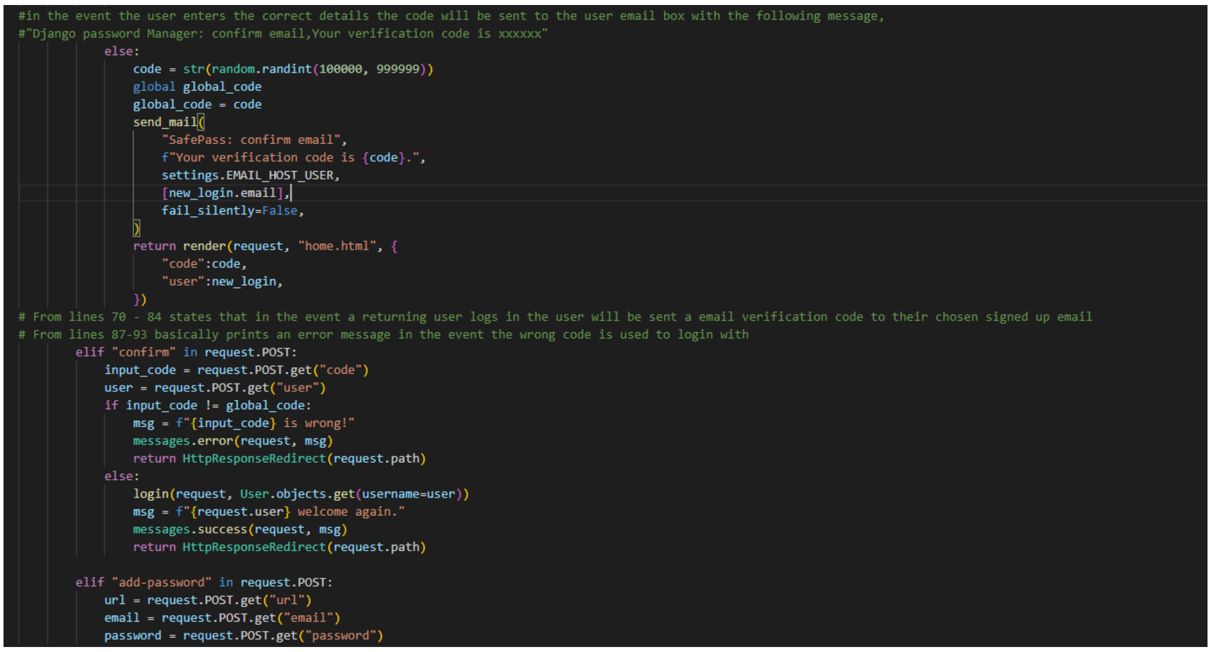
I was gonna attempt to deploy my application live for the final submission but due to my lack of effort and time management skills I failed upon this. Finally, developing SafePass necessitated a mix of web development, database administration, and security abilities. I was able to construct a comprehensive and user-friendly password manager vault application that lets users securely store and manage their login credentials using Django, Python, HTML, CSS, JavaScript, and SQLite.

2.4. Implementation Cont

SafePass is a password management application written in Python and built with the Django framework. It is intended to assist users in securely storing and managing their passwords, removing the need for users to remember various passwords for different accounts.

SafePass password manager vault which is a safe storing location for all of the user's passwords, one of its primary features of my application. I used Django's built-in authentication system to construct a login page that needs Users to enter their credentials before accessing their password vaults to create a vault. In addition, a two-step verification requirement by email confirmation code

email Verification Logic:

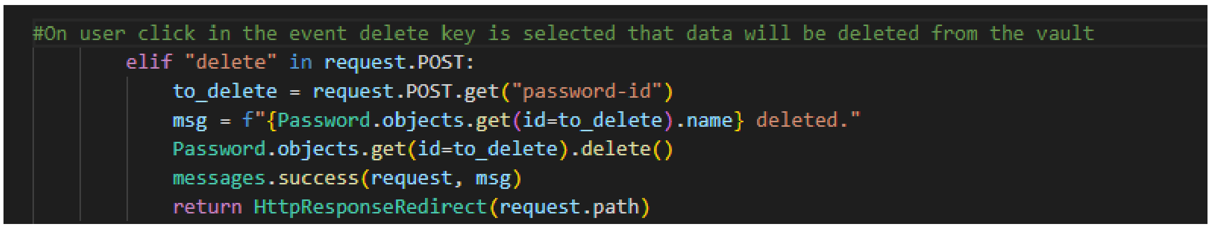


```python
#in the event the user enters the correct details the code will be sent to the user email box with the following message,
#"Django password Manager: confirm email,Your verification code is xxxxxx"
            else:
                code = str(random.randint(100000, 999999))
                global global_code
                global_code = code
                send_mail(
                    "SafePass: confirm email",
                    f"Your verification code is {code}.",
                    settings.EMAIL_HOST_USER,
                    [new_login.email],
                    fail_silently=False,
                )
                return render(request, "home.html", {
                    "code":code,
                    "user":new_login,
                })
# From lines 70 - 84 states that in the event a returning user logs in the user will be sent a email verification code to their chosen signed up email
# From lines 87-93 basically prints an error message in the event the wrong code is used to login with
        elif "confirm" in request.POST:
            input_code = request.POST.get("code")
            user = request.POST.get("user")
            if input_code != global_code:
                msg = f"{input_code} is wrong!"
                messages.error(request, msg)
                return HttpResponseRedirect(request.path)
            else:
                login(request, User.objects.get(username=user))
                msg = f"{request.user} welcome again."
                messages.success(request, msg)
                return HttpResponseRedirect(request.path)

        elif "add-password" in request.POST:
            url = request.POST.get("url")
            email = request.POST.get("email")
            password = request.POST.get("password")
```

As you can see from the comments in the code, the logic which is defined in views.py when a user is logging in an email verification code is sent to their email address, an email verification box is then generated on the screen and awaits for the confirmation code, in the event the user enters an incorrect code the application returns an error to the user stating the code is incorrect and to try logging back in

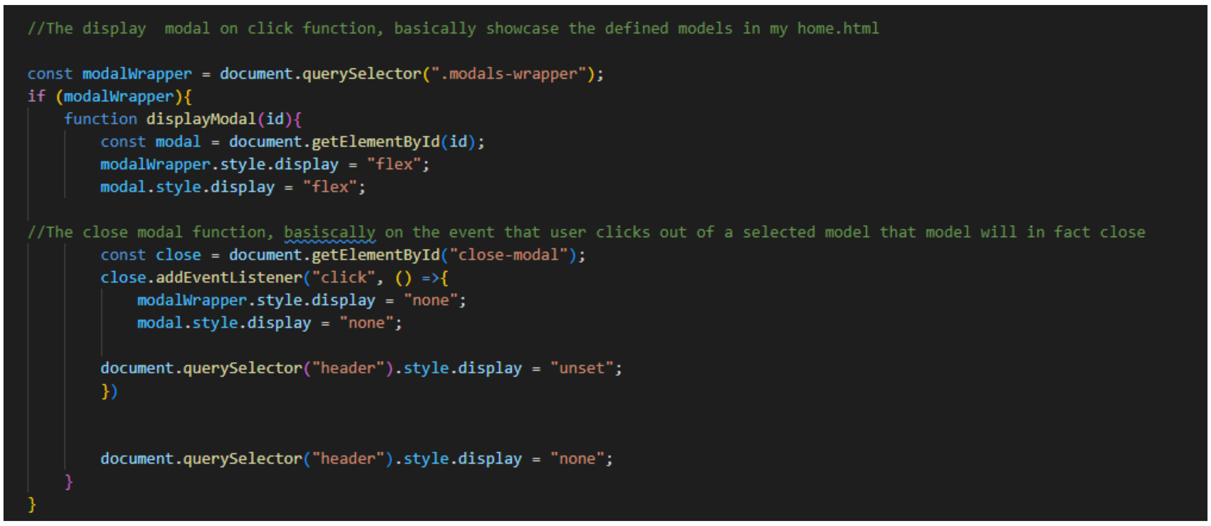
After logging in, a user can create a new vault, update existing vaults, and remove existing vaults. SafePass encrypts passwords and emails before keeping them in the database to ensure the security of the passwords and emails. This implies that even if a hacker has access to the database, they will be unable to read the passwords unless they have the encryption key.

the logic for edit, deleting and adding new Vaults:



```
#On user click in the event delete key is selected that data will be deleted from the vault
        elif "delete" in request.POST:
            to_delete = request.POST.get("password-id")
            msg = f"{Password.objects.get(id=to_delete).name} deleted."
            Password.objects.get(id=to_delete).delete()
            messages.success(request, msg)
            return HttpResponseRedirect(request.path)
```

As you can see from the code snips and their respectful comments these logics are from two separate files within the application but are required within the application, So in the first image which is located in the view.py file the logic defined here states that on the event the user decides to click delete vault that will be executed, an confirmation message will pop up stating what vault has been deleted and the information will be then deleted from the vault

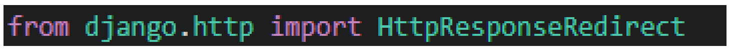


```
//The display  modal on click function, basically showcase the defined models in my home.html

const modalWrapper = document.querySelector(".modals-wrapper");
if (modalWrapper){
    function displayModal(id){
        const modal = document.getElementById(id);
        modalWrapper.style.display = "flex";
        modal.style.display = "flex";

//The close modal function, basiscally on the event that user clicks out of a selected model that model will in fact close
        const close = document.getElementById("close-modal");
        close.addEventListener("click", () =>{
            modalWrapper.style.display = "none";
            modal.style.display = "none";

        document.querySelector("header").style.display = "unset";
        })


        document.querySelector("header").style.display = "none";
    }
}
```

Then here in the second image which is located in the main.js file, this logic allows the vaults to be displayed and to appear with their respective function and stylings set out by the project, without this the vaults will not be viable and data won't have a physical entry point to enter data in.

Note these are what I considered the most important parts of my implementation of this project as they provide the overall initial features that are required to use the application:

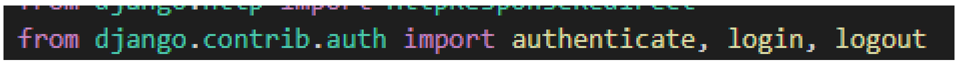
Http Responses



```
from django.http import HttpResponseRedirect
```

This is a Django class that represents a redirect response. It is used to redirect the user's browser to a new URL, generally in reaction to a user activity like as submitting a form or clicking a link.

login:

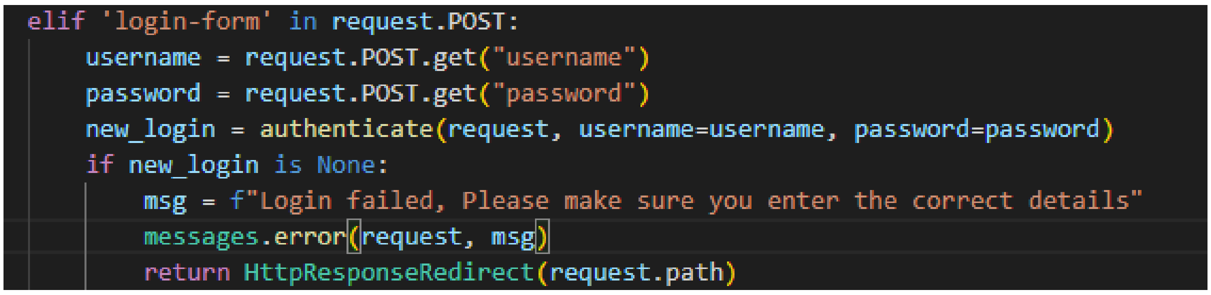


```python
from django.contrib.auth import authenticate, login, logout
```

Django's built-in module **django.contrib.auth** offers a user authentication system. It offers a number of methods and classes for managing user authentication in my case it was used for generating users, logging in users, and validating user permissions.

Off that point though when, importing authenticate, login, and logout from django.contrib.auth imports three methods that are used to manage user authentication in a Django web application such as my own known as SafePass belows they are described as.

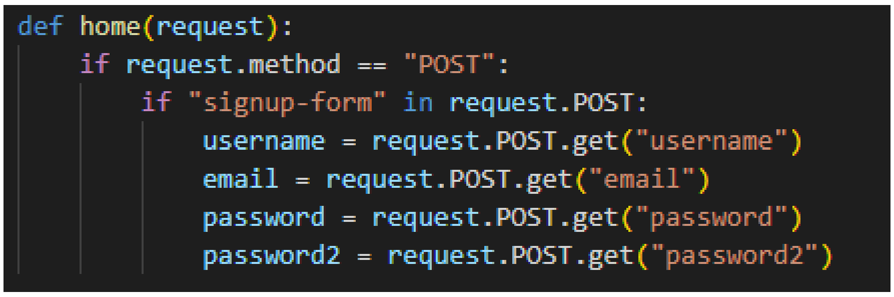
authenticate() is used to check user credentials, login() to log in a user, and logout() to log out the current user.



```python
elif 'login-form' in request.POST:
    username = request.POST.get("username")
    password = request.POST.get("password")
    new_login = authenticate(request, username=username, password=password)
    if new_login is None:
        msg = f"Login failed, Please make sure you enter the correct details"
        messages.error(request, msg)
        return HttpResponseRedirect(request.path)
```

This code is part of a Django view that processes HTTP POST requests. It examines the request for the presence of a parameter named 'login-form,' which "presumably indicates that a user has submitted a login form," and if it does, it attempts to authenticate the user using the specified username and password.

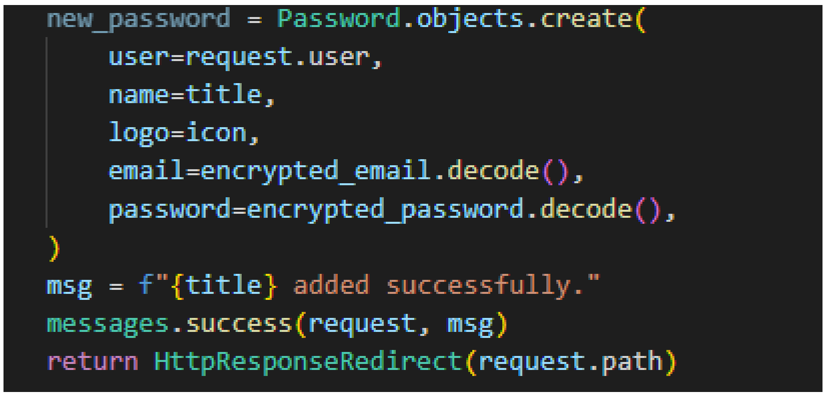
Signup:



```python
def home(request):
    if request.method == "POST":
        if "signup-form" in request.POST:
            username = request.POST.get("username")
            email = request.POST.get("email")
            password = request.POST.get("password")
            password2 = request.POST.get("password2")
```

This is a Django view function that handles HTTP requests to the main page, The code block determines whether the request type is POST which will be indicating that the user has submitted a data-filled form. If the request method is POST, the code block determines if the form supplied was a "signup-form". If this is the case, it uses the request to get the username, email, password, and password2 fields from the form **data.The POST.get()** function returns a value.

Add New Vaults:



```
new_password = Password.objects.create(
    user=request.user,
    name=title,
    logo=icon,
    email=encrypted_email.decode(),
    password=encrypted_password.decode(),
)
msg = f"{title} added successfully."
messages.success(request, msg)
return HttpResponseRedirect(request.path)
```

This block of logic allows the user on the application to create a new Password Vault within the application and then the below fuction message prints a success notification for the user to be aware they created a new vault entry successfully

Final Technology used

Django: a Python web framework for server-side web development.

Python: a general-purpose, high-level programming language used for backend web development.

HTML: a markup language used to create web pages and the interface of the password management vault.

CSS: a style language used to customize the look of the password manager vault interface.

JavaScript: a computer language used to generate interactive components and improve the password manager vault's user experience.

SQLite: is a relational database management system that is used to store user data and login credentials.

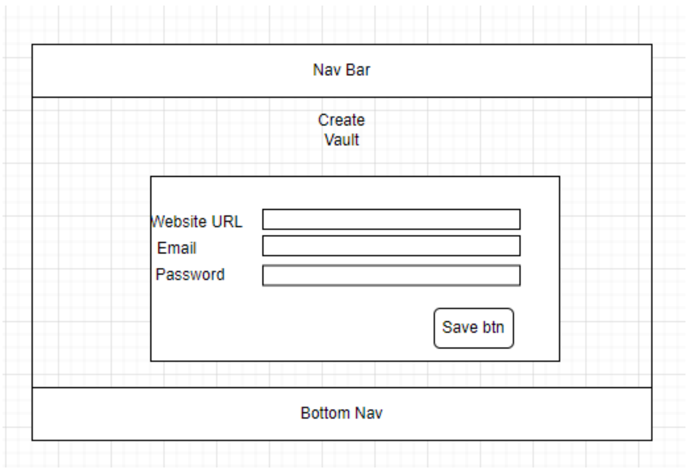
Key derivation: a technique for securely storing users' login credentials that involves generating a unique key from the user's master password.

Cryptography: a library that encrypts and decrypts sensitive data, such as login credentials for users.

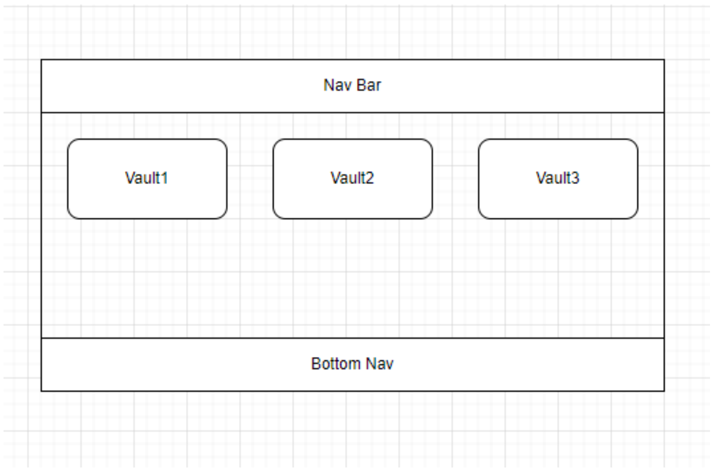
GitHub: Github has been utilized as the application's primary version control system to track code uploads and modifications. It is also used to submit the application's source code.

2.5. Graphical User Interface (GUI)

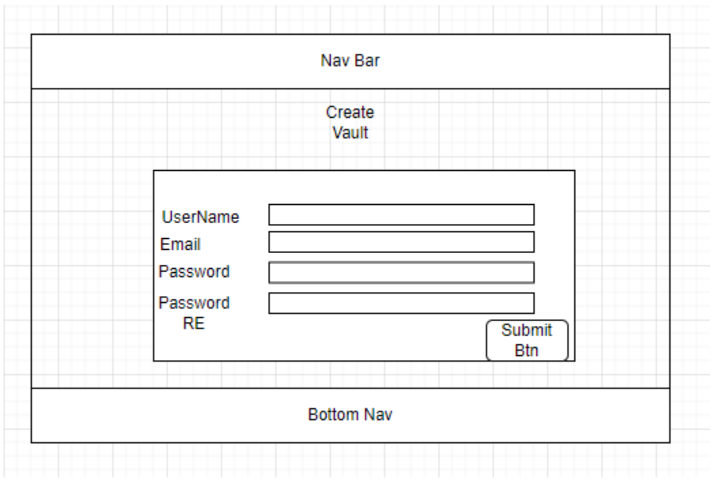
Wireframe Beta Add Vault Page:



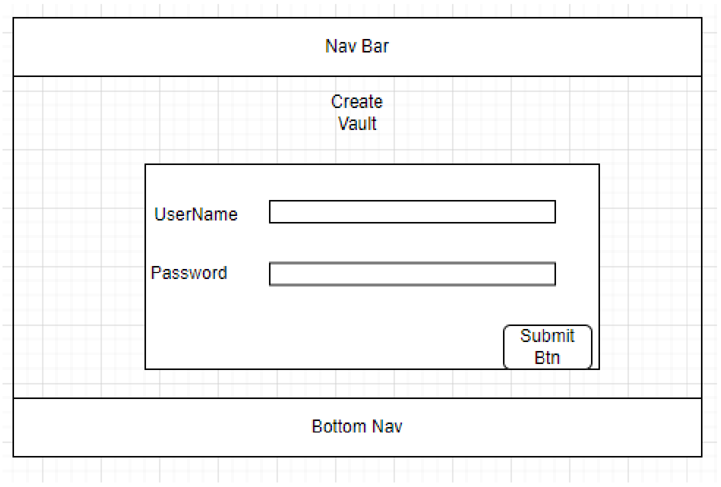
Wireframe beta of the home page:



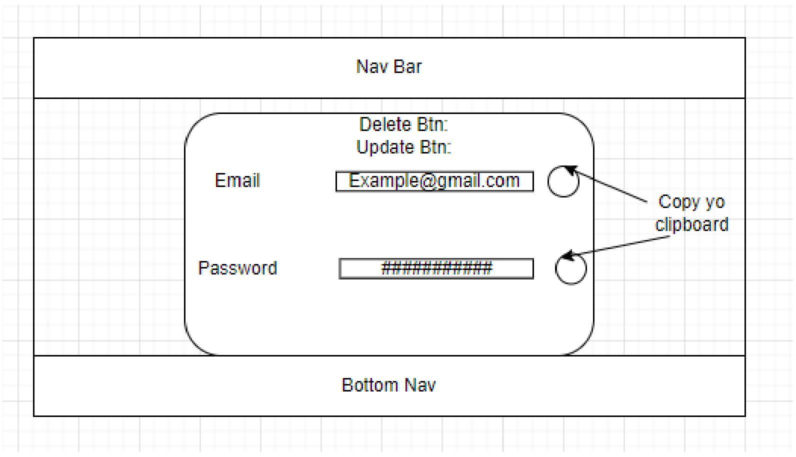
Wireframe Beta of Signup Page:



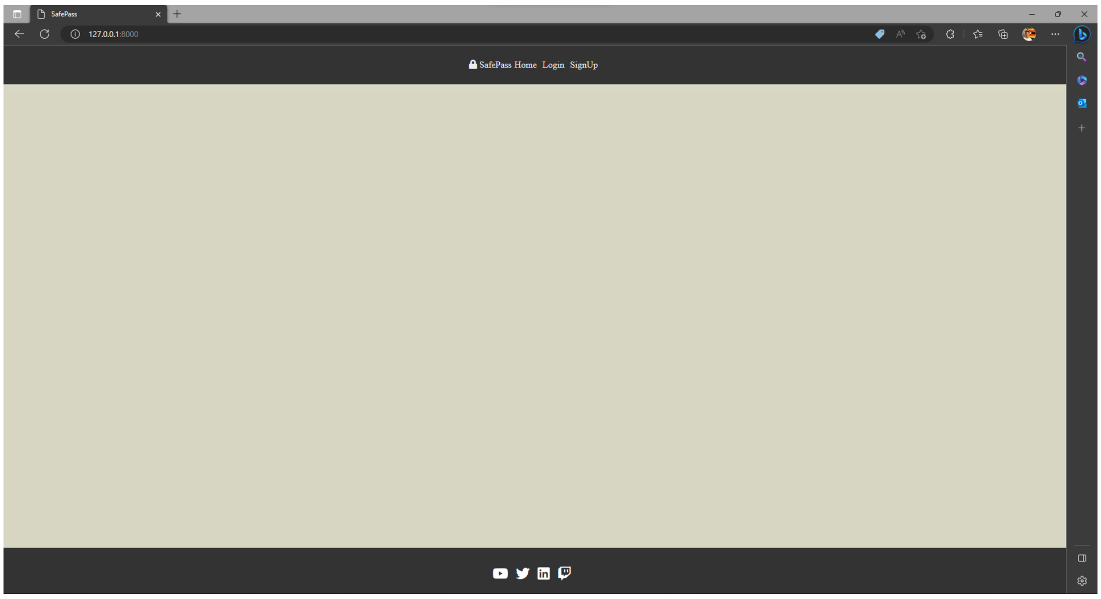
Wireframe Beta of Login Page:



Wireframe Beta of Vault

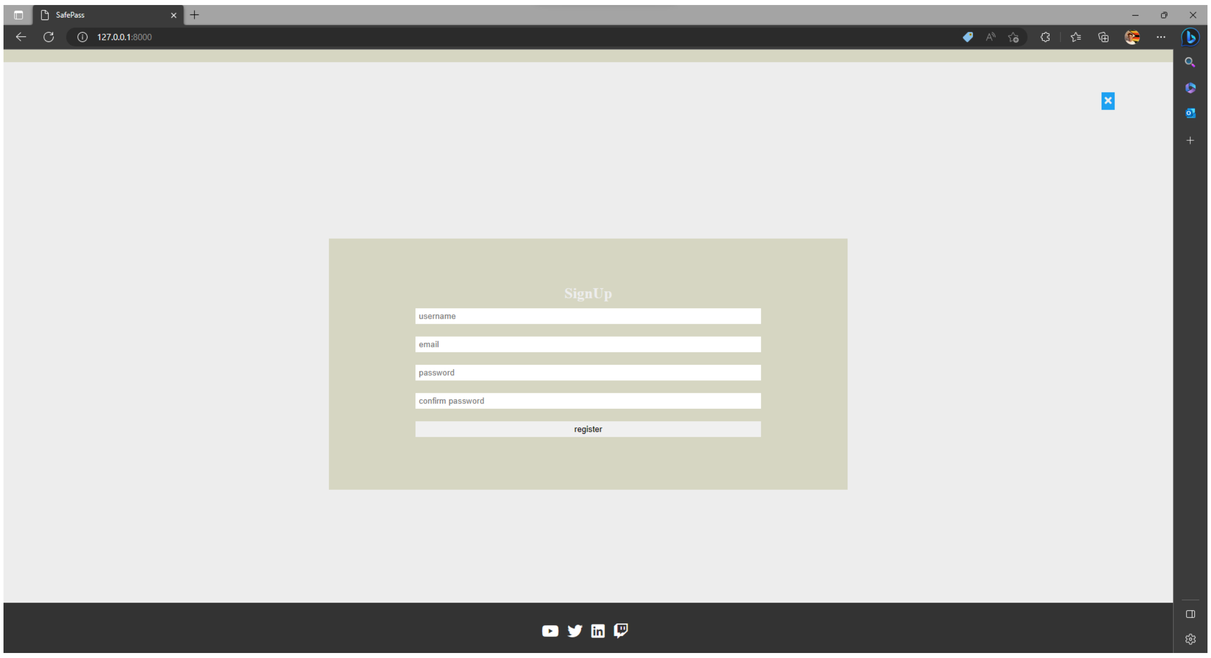


Home Page Final:



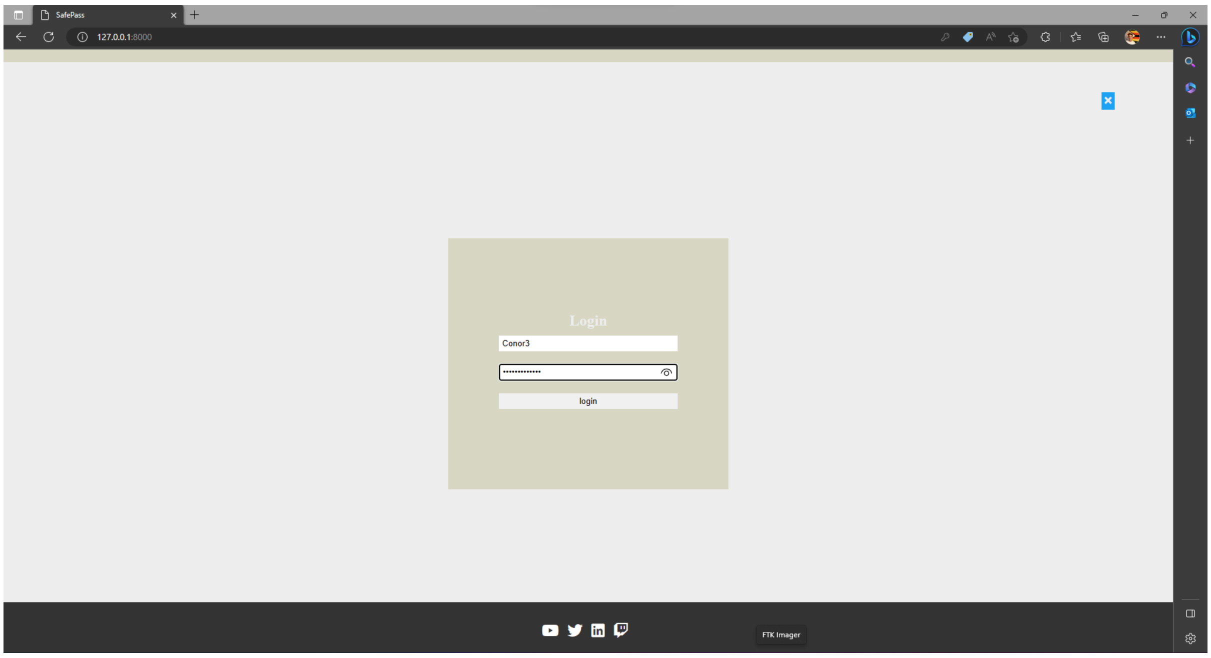
My web application's homepage has a clean and basic design, with a navigation bar at the top that includes options for logging in and joining up. The navigation bar is simple to use and allows users to easily access the application's essential features. For a fun touch, I also included a footer with links to my own social media sites. The general design is designed to be user-friendly and entertaining, with a dash of personality and comedy thrown in for good measure.

Sign Up Page Final:



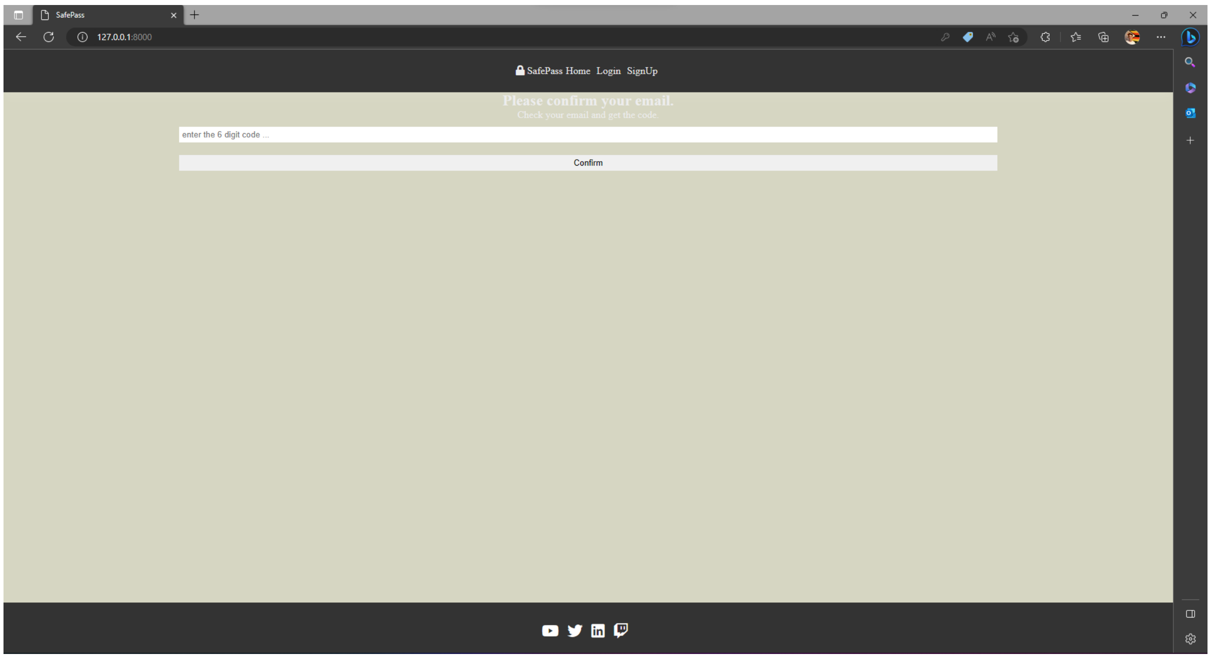
The sign-up screen, which asks users to establish a unique username, provide a valid email address, and create and confirm a password of at least 8 characters. These standards serve to assure the security of user accounts and lay the groundwork for the password management vault's data encryption and security capabilities.

Login Page Final:



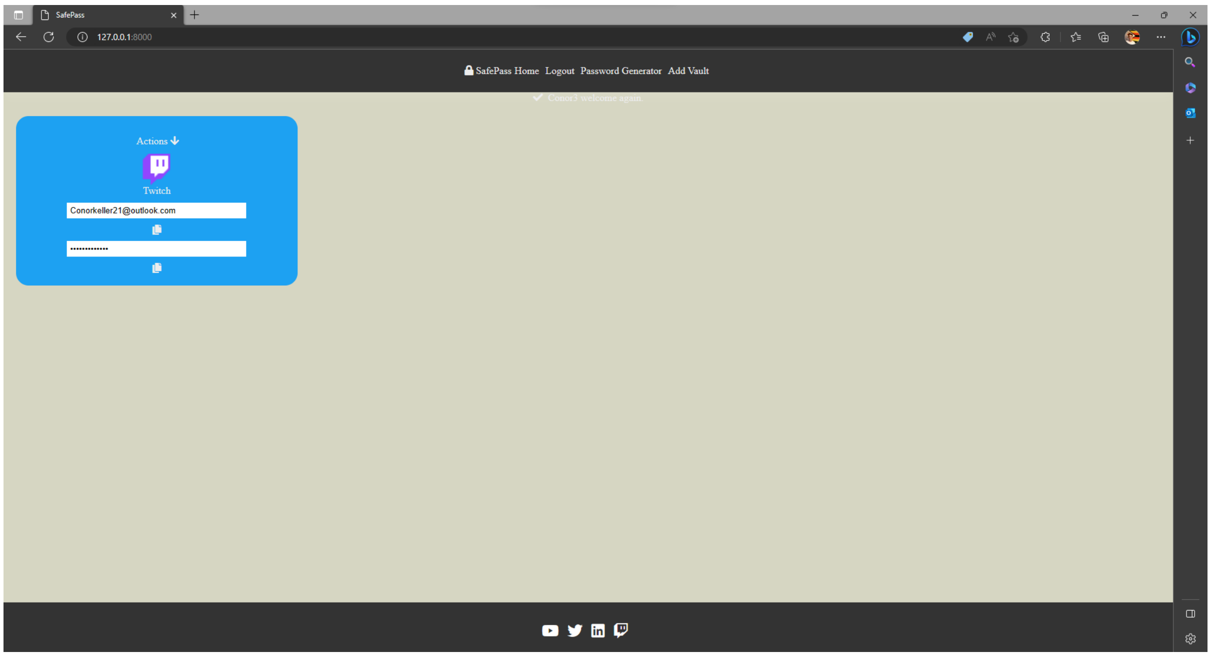
My web application's log-in page has a simple and easy-to-use design, with clear prompts for visitors to input their login credentials.

Email Verification Final:


SafePass
127.0.0.1:8000
SafePass Home Login SignUp
Please confirm your email.
Check your email and get the code
enter the 6 digit code ...
Confirm

I have introduced an email verification option to my web application to improve the security and privacy of user accounts. When a user initially signs in, a verification box appears, prompting the user to confirm their email address by clicking a link given to their registered email account. This step ensures that only authorized individuals may access user accounts and prevents unauthorized access or exploitation of sensitive information. Overall, this email verification option adds an extra layer of security to my password manager vault and gives users more piece of mind.

Successful Login Final:

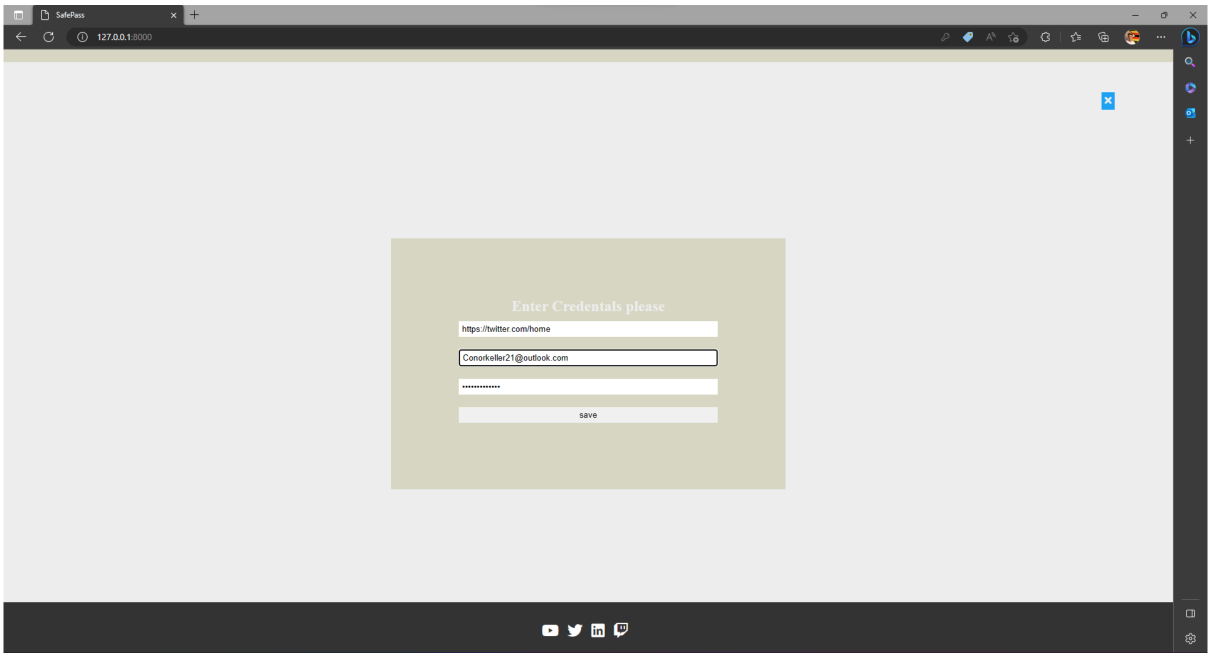


Conor3 welcome again.

Conorkeller21@outlook.com

Displays Successful Login Message upon user logs in and passes the email verification process.

Adding new Vault Final:

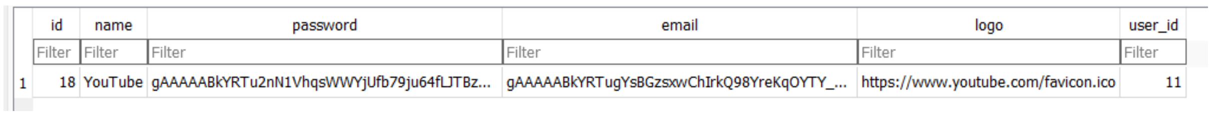


Enter Credentals please

https://twitter.com/home

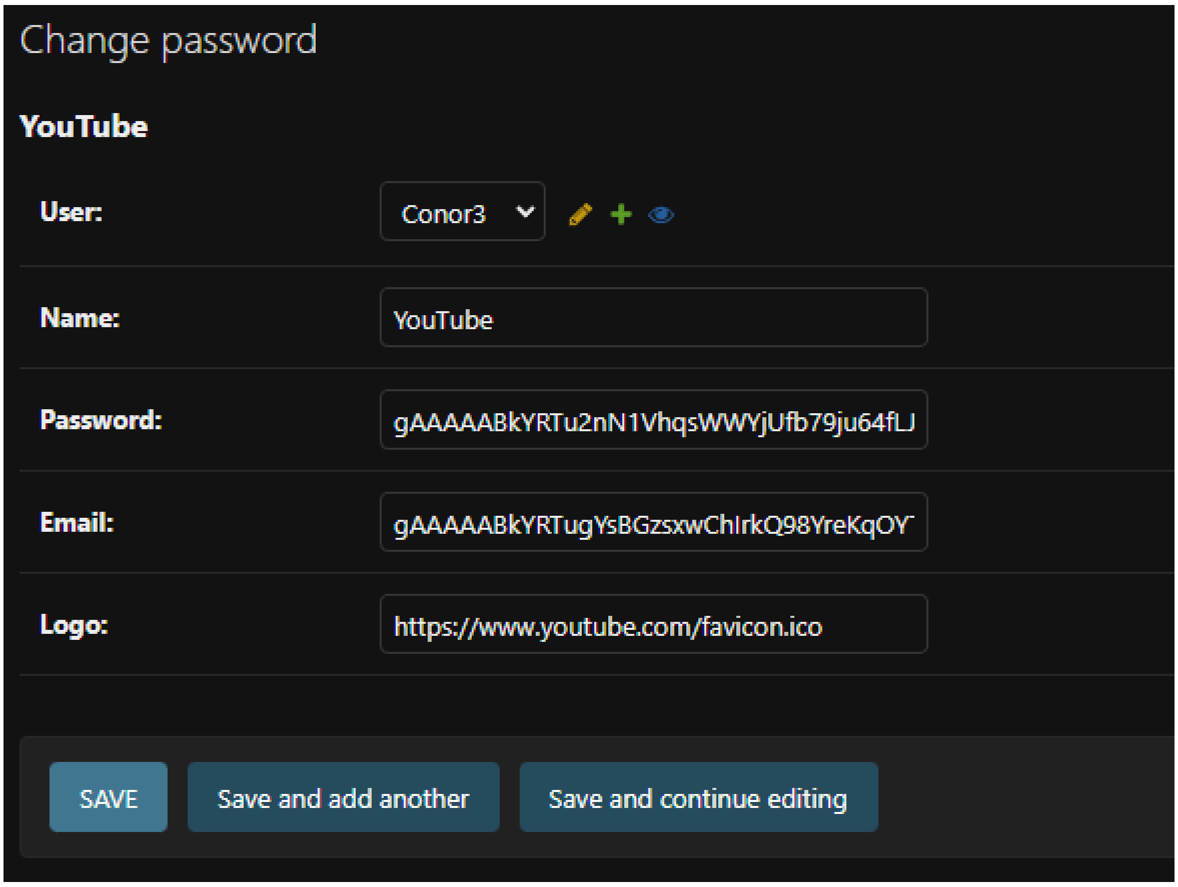
Conorkeller21@outlook.com

The "Add Vault" page of my web application is designed to make it easy for users to store and manage login credentials for various online accounts. To simplify the process of adding new accounts, the page requires users to supply a URL for the site they wish to add. By using the favicon associated with the site, the application automatically obtains the icon and displays it alongside the account name for easy identification. In addition to the URL and account name, users are also prompted to supply the email associated with the account and its password. To ensure the security and privacy of user accounts, the password is encrypted using industry-standard encryption algorithms and stored securely in the SQLite database. Overall, the "Add New Vault" page is designed to be user-friendly, secure, and easy-to-use, helping users to manage their online accounts with greater ease and confidence.



	id	name	password	email	logo	user_id
	Filter	Filter	Filter	Filter	Filter	Filter
1	18	YouTube	gAAAAABkYRTu2nN1VhqsWWYjUfb79ju64fLJTBz...	gAAAAABkYRTugYsBGzsxwChIrkQ98YreKqOYTY_...	https://www.youtube.com/favicon.ico	11

Email and password key viable in the SQLite database and the admin centre

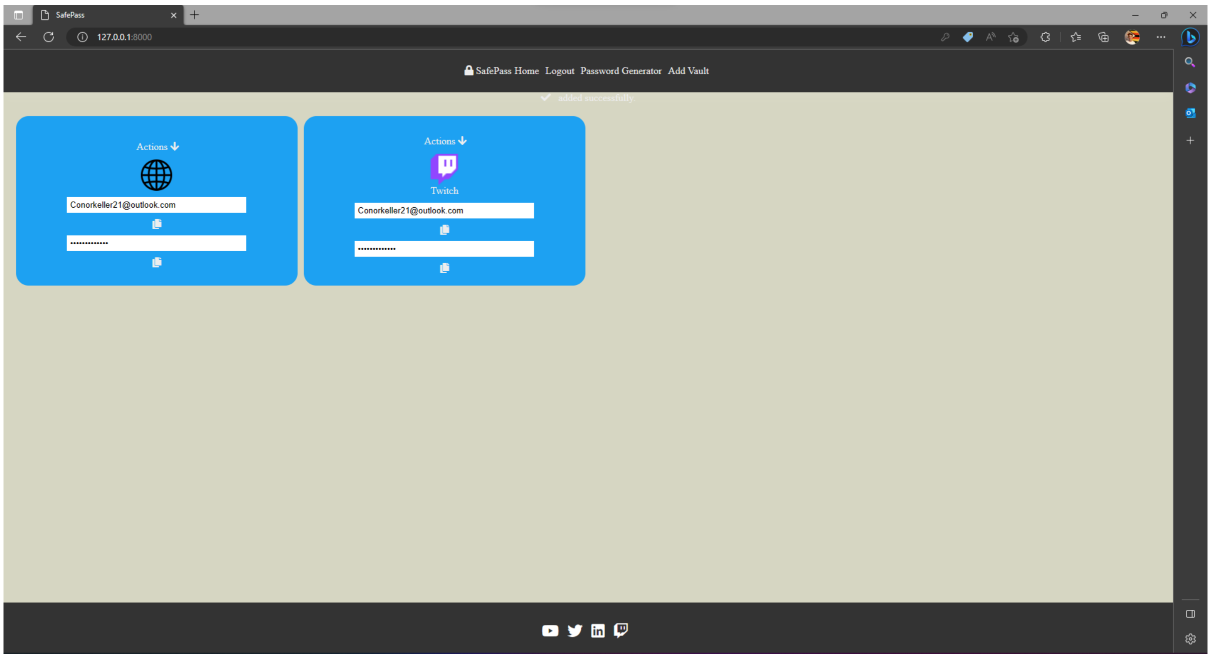


Change password

YouTube

User:	Conor3
Name:	YouTube
Password:	gAAAAABkYRTu2nN1VhqsWWYjUfb79ju64fLJ
Email:	gAAAAABkYRTugYsBGzsxwChIrkQ98YreKqOY
Logo:	https://www.youtube.com/favicon.ico

Successful New Vault Final:



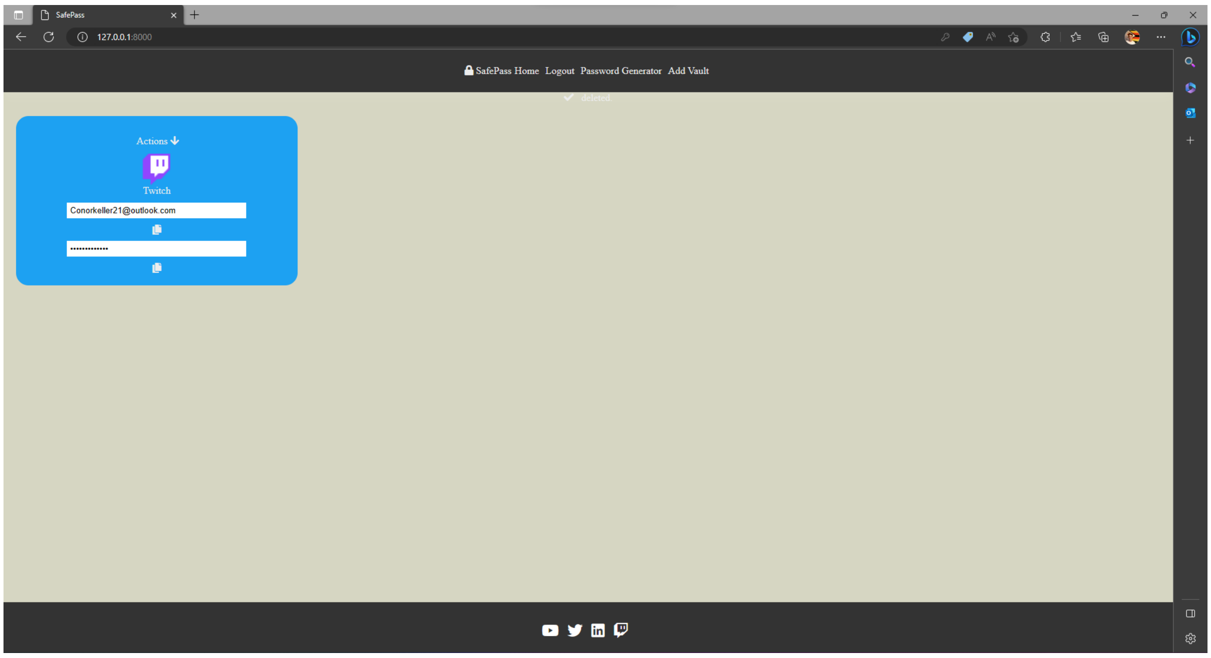
✓ added successfully.

Conorkeller21@outlook.com

Conorkeller21@outlook.com

Proof a new Vault was created.

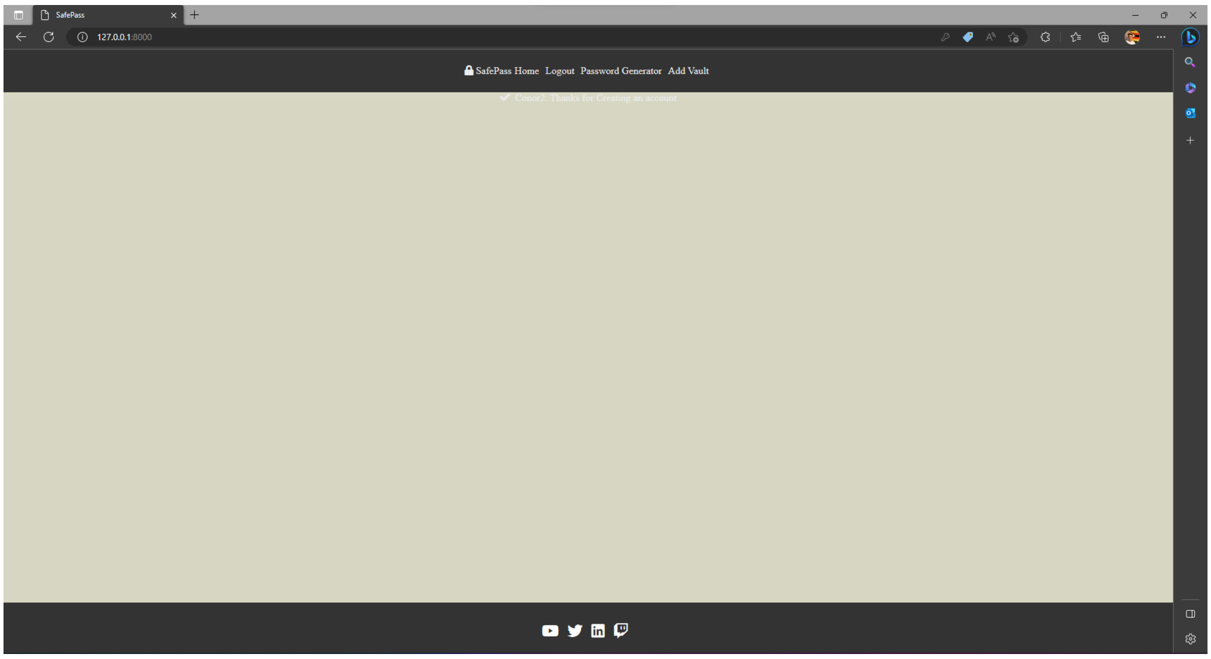
Vault Delete Successfulness Final:



✓ deleted.

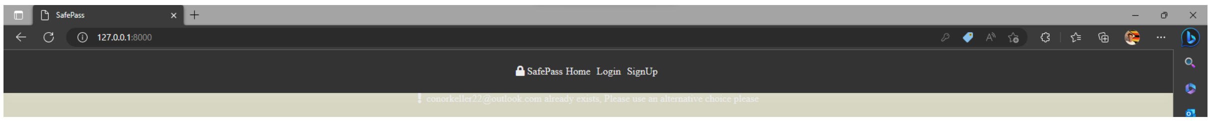
Conorkeller21@outlook.com

Successful Signing up Final:



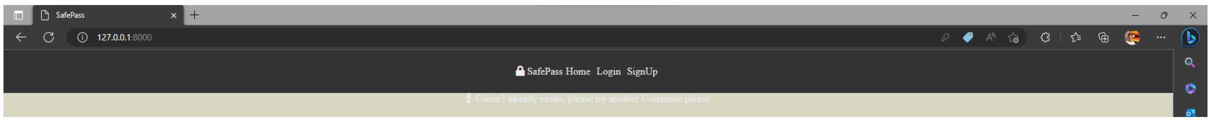
✓ Conor2. Thanks for Creating an account

Fail as User with email already exists:



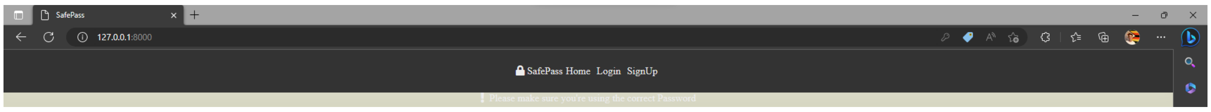
! conorkeller22@outlook.com already exists, Please use an alternative choice please

Fail as User with username already exists:



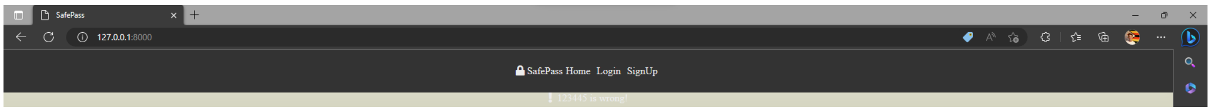
! Conor3 already exists, please try another Username please

Fail as User entered the wrong password twice



! Please make sure you're using the correct Password

Fail as User did not enter the correct email vertifcation code



! 123445 is wrong!

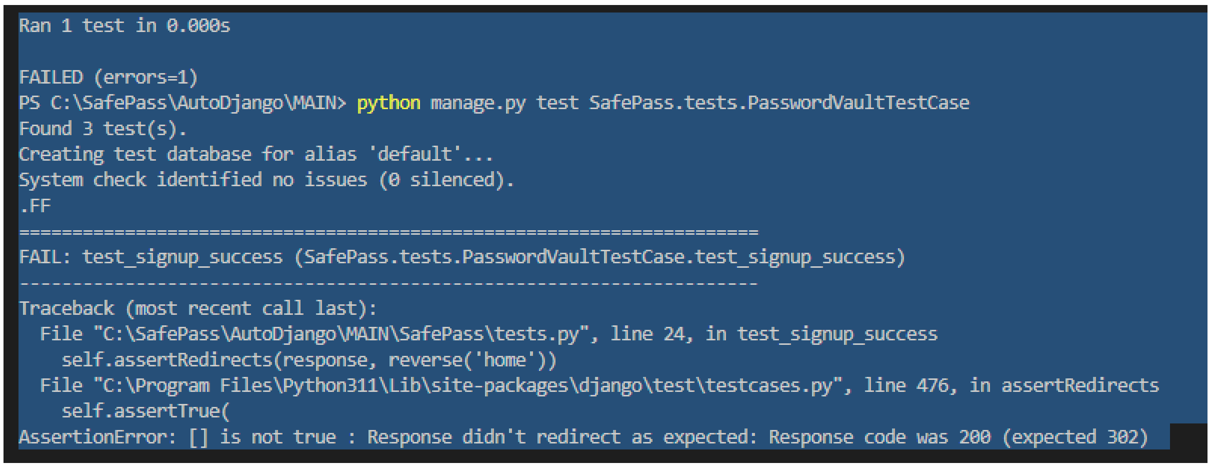
2.6. Testing

3.0 Unit testing:

Test Case ID	T1
Test Component	Sign Up
Test Description	Test the ability for a user registration for the database
Test Priority	1st Priority

Action	**Inputs**	**Expected Result**	**Actual Output**	**Comments**
Step 1: Click the “Sign up” button.	Click	The sign up page successfully loads.	The sign up page successfully loads.	There were no issues during this step.
Step 2: Fill in the required fields.	Text	The user can type in the requested data.	The user was able to type in the requested data.	There were no issues during this step.
Step 3: Click the “Sign Up.	Click	The user’s account will be created and logged in to the main page	The user successfully created the account and is logged in	There were no issues during this step.

How the Tests looked when failed

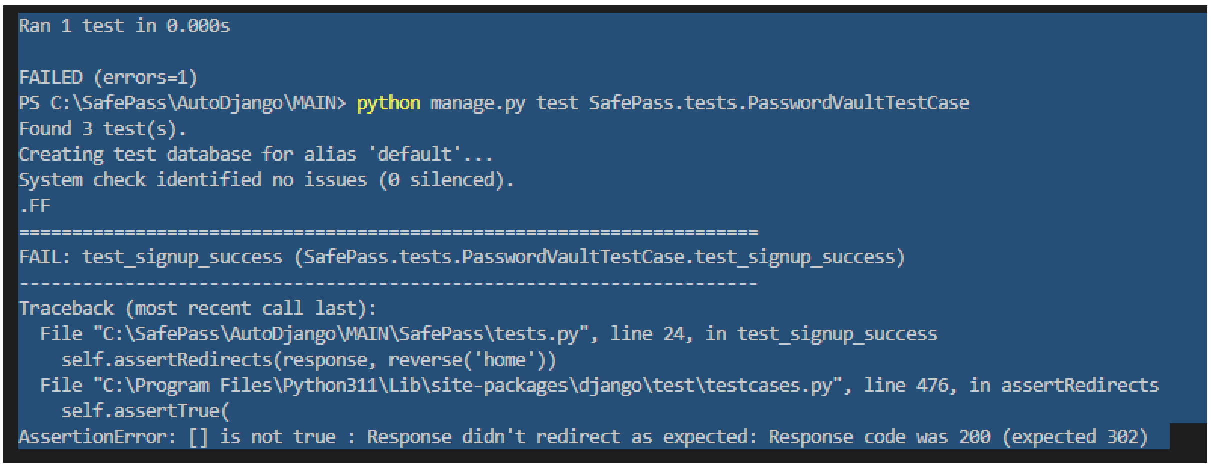


```
Ran 1 test in 0.000s

FAILED (errors=1)
PS C:\SafePass\AutoDjango\MAIN> python manage.py test SafePass.tests.PasswordVaultTestCase
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.FF
======================================================================
FAIL: test_signup_success (SafePass.tests.PasswordVaultTestCase.test_signup_success)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\SafePass\AutoDjango\MAIN\SafePass\tests.py", line 24, in test_signup_success
    self.assertRedirects(response, reverse('home'))
  File "C:\Program Files\Python311\Lib\site-packages\django\test\testcases.py", line 476, in assertRedirects
    self.assertTrue(
AssertionError: [] is not true : Response didn't redirect as expected: Response code was 200 (expected 302)
```

Test Case ID	T2
Teste Component	Login
Test Description	Test the ability for a user login
Test Priority	2nd Priority

Action	Inputs	Expected Result	Actual Output	Comments
Step 1: Click the “Login” button.	Click	The Login page gives error to input correct details	The Login page gives error message	There were no issues during this step.
Step 2: Fill in the required fields.	Text	The user can type in the requested data.	The user was able to type in the requested data.	There were no issues during this step.
Step 3: Click the “Login” button.	Click	The user is logged in and greeted to the main page	The user successfully logged in and greeted with main page	There were no issues during this step.

How the Tests looked when failed

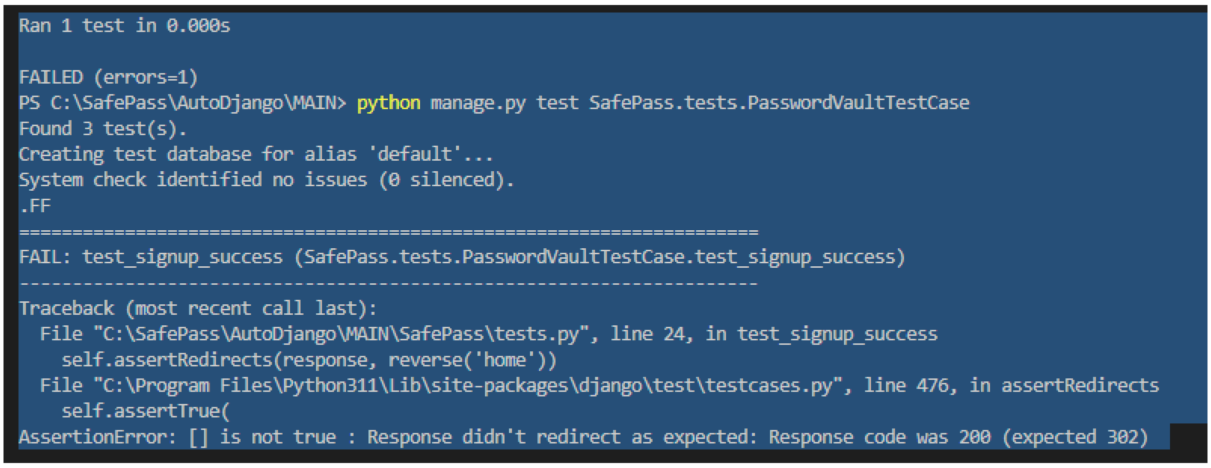


```
Ran 1 test in 0.000s

FAILED (errors=1)
PS C:\SafePass\AutoDjango\MAIN> python manage.py test SafePass.tests.PasswordVaultTestCase
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.FF
======================================================================
FAIL: test_signup_success (SafePass.tests.PasswordVaultTestCase.test_signup_success)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\SafePass\AutoDjango\MAIN\SafePass\tests.py", line 24, in test_signup_success
    self.assertRedirects(response, reverse('home'))
  File "C:\Program Files\Python311\Lib\site-packages\django\test\testcases.py", line 476, in assertRedirects
    self.assertTrue(
AssertionError: [] is not true : Response didn't redirect as expected: Response code was 200 (expected 302)
```

Test Case ID	T3
Teste Component	Add Vault
Test Description	Test the ability for a user to create a new Vault
Test Priority	3rd Priority

Action	**Inputs**	**Expected Result**	**Actual Output**	**Comments**
Step 1: Click the “Add Vault” button.	Click	The Vault page opens up.	The Dashboard page gives error message	There were no issues during this step.
Step 2: Fill in the required fields.	Text	The user can type in the requested data.	The user was able to type in the requested data.	There were no issues during this step.
Step 3: Click the “Save.	Click	The user successfully created a new vault	The user successfully generated the vault and the data was saved	There were no issues during this step.

How the Tests looked when failed

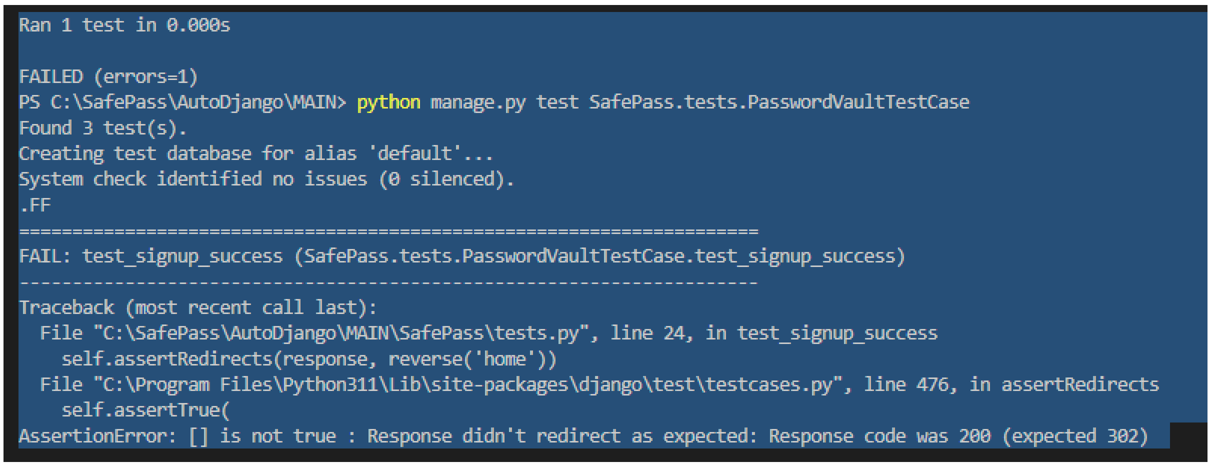


```
Ran 1 test in 0.000s

FAILED (errors=1)
PS C:\SafePass\AutoDjango\MAIN> python manage.py test SafePass.tests.PasswordVaultTestCase
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.FF
======================================================================
FAIL: test_signup_success (SafePass.tests.PasswordVaultTestCase.test_signup_success)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\SafePass\AutoDjango\MAIN\SafePass\tests.py", line 24, in test_signup_success
    self.assertRedirects(response, reverse('home'))
  File "C:\Program Files\Python311\Lib\site-packages\django\test\testcases.py", line 476, in assertRedirects
    self.assertTrue(
AssertionError: [] is not true : Response didn't redirect as expected: Response code was 200 (expected 302)
```

Test Case ID	T4
Teste Component	Delete Vault
Test Description	Test the ability for a user to Delete their Vault
Test Priority	Moderate

Action	**Inputs**	**Expected Result**	**Actual Output**	**Comments**
Step 1: User clicks the Card to delete	Click	The vault page retrieves the data	The Dashboard page outputs the data in the fields	There were no issues during this step.
Step 2: User views “Delete Successful”	Click/Text	Home Pahe will give confirmation message and update	The application Deletes the data from the SQLite Database	There were no issues during this step.

How the Tests looked when failed

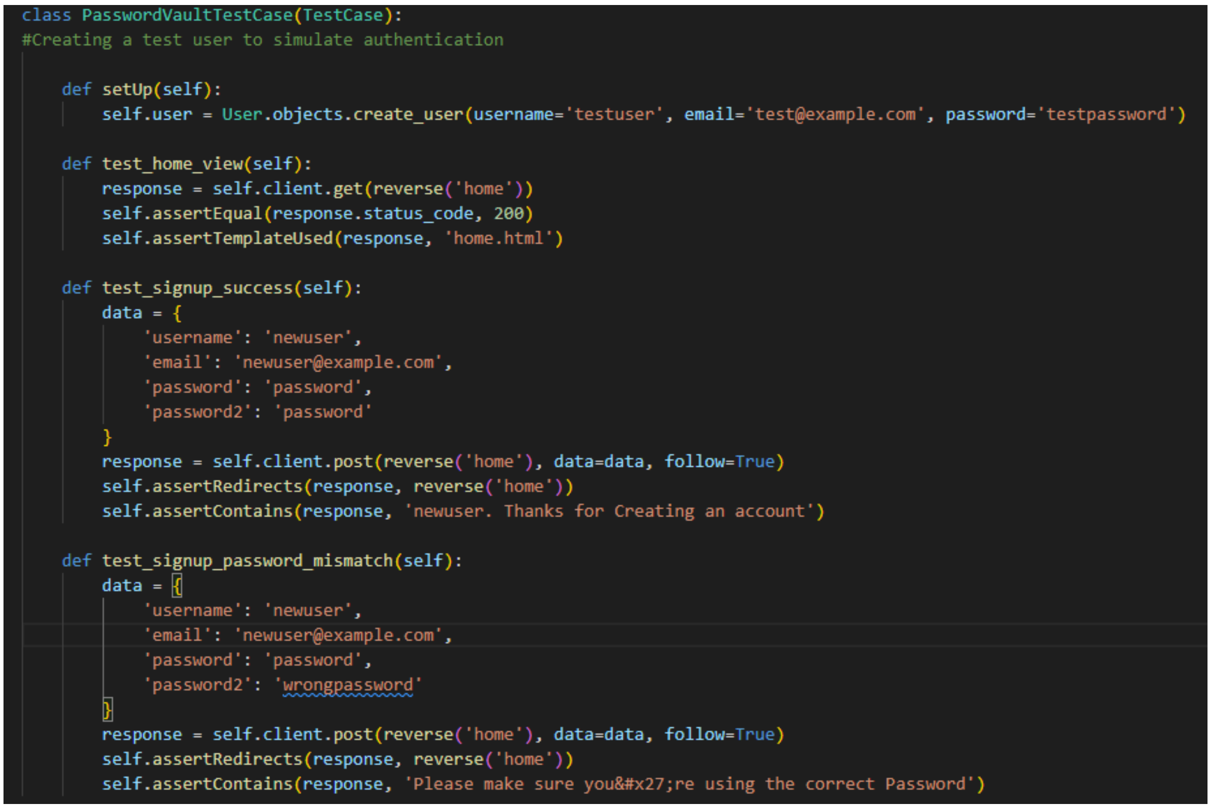


```
Ran 1 test in 0.000s

FAILED (errors=1)
PS C:\SafePass\AutoDjango\MAIN> python manage.py test SafePass.tests.PasswordVaultTestCase
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.FF
======================================================================
FAIL: test_signup_success (SafePass.tests.PasswordVaultTestCase.test_signup_success)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\SafePass\AutoDjango\MAIN\SafePass\tests.py", line 24, in test_signup_success
    self.assertRedirects(response, reverse('home'))
  File "C:\Program Files\Python311\Lib\site-packages\django\test\testcases.py", line 476, in assertRedirects
    self.assertTrue(
AssertionError: [] is not true : Response didn't redirect as expected: Response code was 200 (expected 302)
```

4.0 Integration Testing:

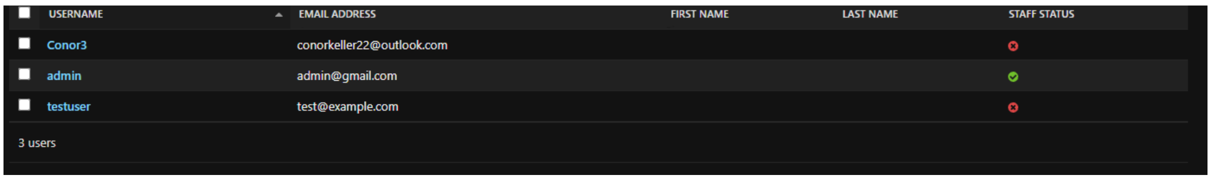
Following on from unit testing on the primary application functionality and combining all of the components, the following findings were obtained:

The following code was ran through the application (To sign up with an tests account)in the testing file and this is what the function was



```
class PasswordVaultTestCase(TestCase):
#Creating a test user to simulate authentication

    def setUp(self):
        self.user = User.objects.create_user(username='testuser', email='test@example.com', password='testpassword')

    def test_home_view(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'home.html')

    def test_signup_success(self):
        data = {
            'username': 'newuser',
            'email': 'newuser@example.com',
            'password': 'password',
            'password2': 'password'
        }
        response = self.client.post(reverse('home'), data=data, follow=True)
        self.assertRedirects(response, reverse('home'))
        self.assertContains(response, 'newuser. Thanks for Creating an account')

    def test_signup_password_mismatch(self):
        data = {
            'username': 'newuser',
            'email': 'newuser@example.com',
            'password': 'password',
            'password2': 'wrongpassword'
        }
        response = self.client.post(reverse('home'), data=data, follow=True)
        self.assertRedirects(response, reverse('home'))
        self.assertContains(response, 'Please make sure you&#x27;re using the correct Password')
```

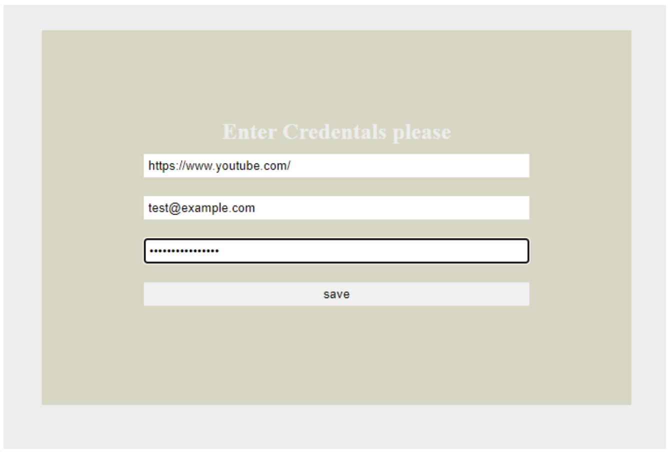
Below you see the TestUser email account was generated



USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
Conor3	conorkeller22@outlook.com			✗
admin	admin@gmail.com			✓
testuser	test@example.com			✗

3 users

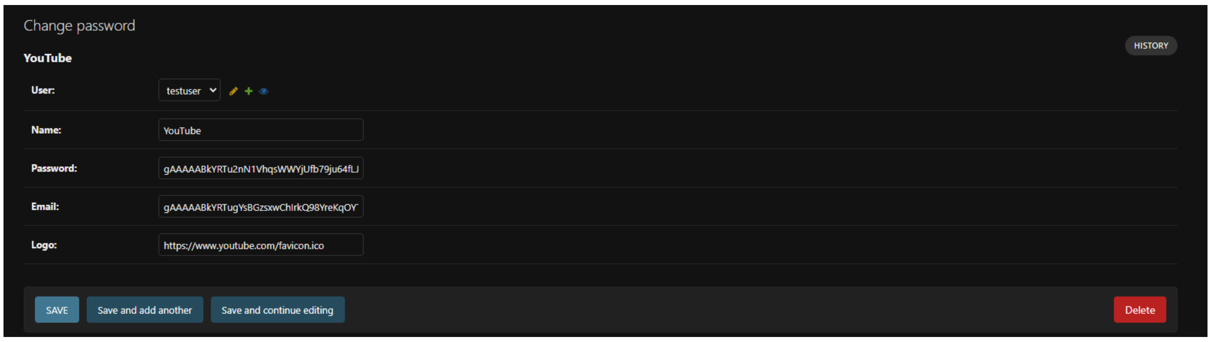
I Then Logged into that test account and manually added a youtube vault manually



Enter Credentals please

https://www.youtube.com/
test@example.com
••••••••••••••••

Then by checking the admin profile online I can check the entered data from the test account like so


Change password

YouTube

User: testuser

Name: YouTube

Password: gAAAAABkYRTu2nN1VhqsWWYjUfb79ju64fLJ

Email: gAAAAABkYRTugYsBGzsxwChlrkQ98YreKqOY

Logo: https://www.youtube.com/favicon.ico

This showcases that the test user was able to create a vault and a key id was given to the password and email address used in its creation.

5.0 Acceptance Testing:

From doing the Unit Testing and Integration testing I was able to conclude that the application works the way it has been intended to without failure the results as follows below were created.

Test for successful sign up:

Given that the user is on the homepage

When the user enters valid details in the signup form and submits

Then the user should be redirected to the homepage

And a success message should be displayed

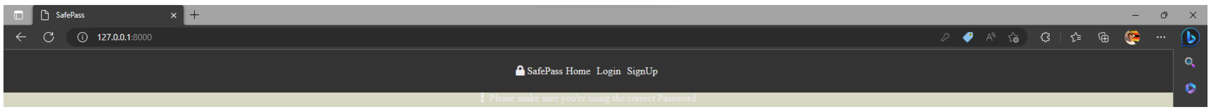
Test for failed sign up due to mismatched passwords:

Given that the user is on the homepage

When the user enters valid details in the signup form but enters mismatched passwords

Then the user should be redirected to the homepage

And an error message should be displayed indicating the issue


❗ Please make sure you're using the correct Password

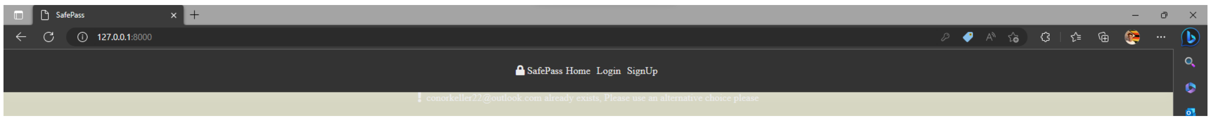
Test for failed sign up due to existing email:

Given that the user is on the homepage

When the user enters an email that already exists in the database in the signup form and submits

Then the user should be redirected to the homepage

And an error message should be displayed indicating the issue



! conorkeller22@outlook.com already exists, Please use an alternative choice please

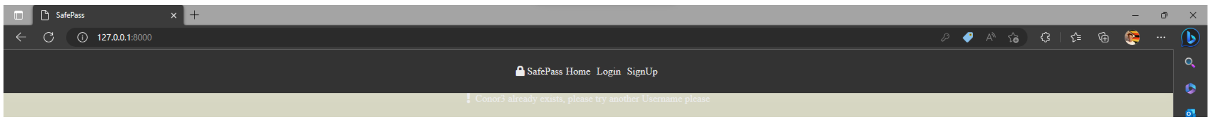
Test for failed sign up due to existing username:

Given that the user is on the homepage

When the user enters a username that already exists in the database in the signup form and submits

Then the user should be redirected to the homepage

And an error message should be displayed indicating the issue



! Conor3 already exists, please try another Username please

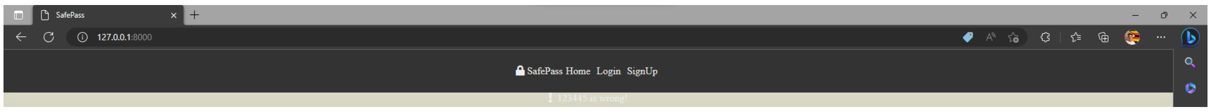
Test for failed email confirmation due to incorrect verification code:

Given that the user is on the homepage and has logged in

When the user enters an incorrect verification code in the confirmation form and submits

Then the user should be redirected to the homepage

And an error message should be displayed indicating the issue



! 123445 is wrong!

5.1. Evaluation

Creating a Django-based password management vault with login/signup functionality and email verification was a difficult assignment for me. Django is a sophisticated web framework with a plethora of features that may be utilized to create a complex application. However When I began the task of incorporating these aspects into a unified project it became time-consuming.

One of the most difficult aspects of developing a password management vault is maintaining the security of user data. Passwords are sensitive data that must be encrypted and securely maintained. Django includes a number of tools for enabling safe authentication, including a built-in User model that may be modified to incorporate extra features and functionality. Email verification is also essential for ensuring that only authorized users have access to the system.

Designing the user interface is another difficulty. The password management vault should be simple to use and straightforward for users, while also offering enough security. This necessitates careful consideration of the user experience and the creation of a practical and aesthetically pleasant interface.

The application's testing and debugging are also important challenges. Django includes a powerful testing framework that can help ensure that the application is functioning properly. However, testing and debugging can still be time-consuming, especially if the application's components interact in complicated ways.

Overall building a Django-based password manager vault with login/signup capabilities and email verification is a difficult endeavour that necessitates careful thought, design and execution.

6.0 Conclusions

To summarize, developing a Django-based password manager vault with login/signup capabilities and email verification is a difficult operation that necessitates attention to detail and a thorough understanding of the Django framework. Security, user experience, testing, debugging, and deployment are all key components of the development process that must be properly considered.

7.0 Further Development or Research

8.0 References

Author/Uploader: 3Blue1Brown

Year: 2018

Title: But what is a Neural Network? | Deep learning, chapter 1

Publisher: YouTube

URL: https://www.youtube.com/watch?v=rHux0gMZ3Eg&t=1673s

Author/Uploader: w3schools.com

Year: 2021

Title: Django Tutorial

Website name: w3schools.com

URL: https://www.w3schools.com/django/

Author/Uploader: CrashCourse

Year: 2018

Title: Natural Language Processing: Crash Course Computer Science #36

Publisher: YouTube

URL: https://www.youtube.com/watch?v=Um-rWZKhL3E

Author/Uploader: freeCodeCamp.org

Year: 2019

Title: Full Python Programming Course (Basics, OOP, Modules, PyQt)

Publisher: YouTube

URL: https://www.youtube.com/watch?v=cu-1EdSMzK0

(usabilityfirst, n.d.)

9.0 Appendices

See Project Proposal (Needs to be amended over Christmas break)

9.1. Project Proposal

National College of Ireland

National College of Ireland

Project Proposal

< The Key(Working Title)>

25/10/22

<Computing Project >

<Cyber Security>

<Academic Year 2022/2023>

<Conor Keller>

<x18353923>

< x18353923@student.ncirl.ie>

Contents

Objectives

The objective of this project is to create **a secure password management system** that **is web**-**based**, of which will allow **non-technical users** to store their password in a safe secure location on any given machine. The web application should provide the following features**, saving passwords**, **random generation password creation, allow users to set password to their own criteria**, **master account creation of up to three users**, **two factor authentication**, **Face ID authentication** & **browser extension functionality.** Along with these functions I intend for the application to be available to users via **PC**, **Tablet** & **Mobile**.

On the initial start-up of the application the users should be allowed to create a Master Admin account by suppling an email, Password, Phone number for two step verification along with enabling a webcam/Phone camera to allow the face ID feature to take notice of how the user appears, once this initial login procedure has completed the given should easily be allowed to cache their information to the application.

The Application should have all the above features working in unison with seamless connectivity, once a user has supplied their login details such as the email & password that user should be able to cache those details into the application and gain access to those cached items via logging in with their Admin credentials, once done the application portal should showcase the email/password along with the given name of the cached detail ex Meta Login, PlayStation Login, Steam etc.

Background

My choice in regard to the project is a simple answer, the most common project that was created last year by cyber security students was a web-based password manager application, by knowing myself and my capabilities I chose to create my own version of the same basic idea as my primary objective of this module is to create an application I can call my own and be proud of while minimising the stress on myself by coming up with an original idea that could lead to a lot of wasted time and energy on my part.

Through my investigation on this topic I have come across a wide array of "Password Safes" on the internet and a feature that seems to be missing in most of these applications seems to be that the Admin Master accounts have a security flaw within them which is in the instance that a malleolus hacker has gained access to the given users Master Account due to the main users using a common password they may use universally, within my application I intend to have a feature where the Master Admin Account would require the user to change their Master Password every twenty one days, in doing so we prevent outside sources gaining access to the application.

On how I will achieve the objects listed above will be through my own research via GitHub, Stack Overflow & YouTube tutorials as this will be the first time I am taking on a task as high as this one, a lot of practice and trial & error will be required to complete the objects. As I am not mostly familiar with the new technologies I will be using on this project.

State of the Art

As mentioned prior my idea is not an original one and frankly may not be the only Password manager application in development among the cyber students, for example the following project titles last year were password managers, CyberVaut, Dragon Pass, Vault Knox Password Manger & The Ace Manager. It seems to be a popular choice among cyber students, so it was only wise of me to tackle the same tasks and possible improve upon the ideas that were presented last year.

The unique selling point of my version of the Password manager application is that I plan to allow the application to be used via PC, Tablet & Mobile as I noticed from my research via the NCI showcase site none of the application mentioned a mobile or tablet functionality via their project description all of them primarily focused on the PC side of things, so I plan to have my application be compatible with those three mediums with seamless transition among the devices.

A popular application like my own I found online is known as “LastPass”.

For most used internet browsers, including Google Chrome, Firefox, Edge, and Safari, there is an extension called LastPass. The extension works across browsers and logs into your password "vault" using your email address and a master password. For the website you want to save your information for, LastPass can create or remember your own password for their vault. Your information can be filled up automatically without you having to touch a button or type anything. In addition to storing login information, LastPass also lets users store payment information in a vault that functions similarly to a password vault with autofill features.

Technical Approach

I have chosen the waterfall development approach for this project due to its simplicity, with the waterfall methodology I can breakdown the process into sequential phases focusing on my primary goals. Each phase in this project must be 100% completed before I can move on to the next phase, with this methodology I can knock each task down the ladder with confidence and I don’t have to go back and re do the phase. The pro of this form is that it’ll make it easier for me to manage and plan my objectives clearly with a step-by-step goal list I can follow in an easy manner.

The **Requirements** within the project depend on the belief that I have outline a clear cohesive plan this includes, research on the languages I intend to use, libraries and any other type of technology ill require for the application, when doing my research I will be using the likes of GitHub, Stack Overflow & YouTube, this is the most important stage within the project as this will set out the course of the whole devotement on this project.

The next step will be **Design,** this will be where I design my technical solution based upon my research from the requirement phase, this will include data models, layouts & application requirements in doing this process I will be able to identify the scope of the project.

Once design is completed, I will move onto the **Implementation,** this should be the shortest phase as my research and design have been completed which allows me to move onto the programming of the application based upon what was completed in the previous phases, if in the instance changes will need to be made to my application, I will have to go back onto the design phase and rethink my approach.

After I have completed my implementation, **Testing** will be the primary focus to ensure my completed application functions as intended, this can be done by using a select pool of users to test out the application and all the features available, this data will need to be recorded and showcased to show proof of functions not working as intended.

Once the previous four phases are completed ill move onto the final phase which is **Deployment/Maintenance,** this is where the application has been completed and submitted in a live status, where users of all kinds can use the application and supply me with a bigger pool of data to determine the application functions as intended.

Technical Details

From my current research the following languages are what I intend to use, these languages are new to me so I will require a lot of practice and understanding before I can develop the application itself but as follows these are the languages, I will use to create my application. Next.js, React.js, Node,js, Fasitfy, TypeScript & crypto.js, I have some familiarity with these Languages having used them in previous college projects but to use them all together in conjunction to create my application will be anew task in itself for me.

My main text for this application will be Microsoft visual studio as I have been using this editor for the past two years, so I feel comfortable and confidence using this in the creation of the application, plus the benefits of using the above-mentioned languages can be used within the VsCode as it provides a simplicity and a smooth experience.

I will be using MongoDB as my database for this application, from my current research it seems to be a popular choice for this type of application, I am not complete familiar with this type of DB so I will have to do more research and watch more tutorials on how to proceed with using this in conjunction with my chosen languages.

From my current research the following algorithms that are required to complete this application are as follows, just a side note I am not familiar with these at all as I have not needed to use any form of algorithms in previous projects so this will be a taxing process on myself, but the result will showcase a sound project. SHA256 - Predictable password hash, Argon2 - Unpredictable password hash for the database, pbkdf2 - Generate the vault key &AES256 - Encrypt and decrypt the vault.

Special Resources Required

As of the moment I'm composing my proposal, I already have access to all the materials I'll need, thus I won't require any additional special resources to do my project. However, I will need to schedule meetings with my supervisor to help me and direct me while I work on the project

Project Plan

I plan to spread out all my work across the year while respecting other submissions like the Midpoint and my other modules as the deadline for the Software Project's final submission is sometime in late May. As of the time that I am writing this proposal, my project is still in the research stage. I'm still deciding what I want to implement, how I can go about accomplishing it, and what I can't get to work the way I expect it to and how to find workarounds.

As of 26th of October I have been given my mentor who I have since contacted to touch base and discuss the scope of my project and in doing so I will be able to gain some clarity on how I shall proceed with the project, I expect to have a meeting with my mentor within the next one to two weeks as of 27th of October I have not heard back from my mentor, within the meeting its self I want to go over my proposal with them and hopefully through this meeting I can gain clarity and a guide on how to proceed as a guiding hand will be a big help for me.

I am using my technical approach guide within my project plan as the information previously said is indeed how I plan to proceed with this project.

The **Requirements** within the project depend on the belief that I have outline a clear cohesive plan this includes, research on the languages I intend to use, libraries and any other type of technology ill require for the application, when doing my research I will be using the likes of GitHub, Stack Overflow & YouTube, this is the most important stage within the project as this will set out the course of the whole devotement on this project.

The next step will be **Design,** this will be where I design my technical solution based upon my research from the requirement phase, this will include data models, layouts & application requirements in doing this process I will be able to identify the scope of the project.

Once design is completed, I will move onto the **Implementation,** this should be the shortest phase as my research and design have been completed which allows me to move onto the programming of the application based upon what was completed in the previous phases, if in the instance changes will need to be made to my application, I will have to go back onto the design phase and rethink my approach.

After I have completed my implementation, **Testing** will be the primary focus to ensure my completed application functions as intended, this can be done by using a select pool of users to test out the application and all the features available, this data will need to be recorded and showcased to show proof of functions not working as intended.

Once the previous four phases are completed ill move onto the final phase which is **Deployment/Maintenance,** this is where the application has been completed and submitted in a live status, where users of all kinds can use the application and supply me with a bigger pool of data to determine the application functions as intended.

Time frames and dates will be decided once I have met up with mentor as the scope of the project is to be determined as I am currently still in the research phase.

- Project Pitch Video October 9th Video URL to Moodle
- Project Proposal October 30th   Moodle Turnitin
- Project Ethics Form October 30th (Monthly) Moodle - Notify Supervisor of submission
- Mid-Point Implementation, Documentation & Video Presentation December 20th Video URL, Document & Slides to Moodle
- Final Implementation, Documentation & Video Presentation May 14th Moodle Turnitin

- Viva Examination (By exception only) May 17th - 19th (By exception only) MS Teams
- Project Showcase May 31st (TBC) TBC

Testing

Regarding the testing of my application there will be four parts as part of the standard testing rubric, this will include **Unit Testing, Integration Testing**, **System Testing** & **Acceptance testing**.

Unit Testing: During this initial stage of testing, the program is put through evaluations that concentrate on sections or elements of the software to see if each one is fully functional. This project's primary goal is to assess whether the application performs as intended.

Integration Testing: Individuals have the chance to combine and test every component of a software during integration testing. The goal of this testing level is to identify interface flaws between the modules and functions. This is especially useful because it shows how well the units are coordinating their operations.

System Testing: The first level of testing that is done on the entire application is system testing. This level's objective is to assess the system's compliance with all of the listed requirements and ensure that it satisfies the Quality Standards. Independent testers who weren't involved in the program's development perform system testing.

Acceptance Testing: To ascertain whether the system is prepared for release, acceptance testing, also known as user acceptance testing, is performed. Requirement's modifications might occasionally be misread during the software development life cycle in a way that does not satisfy the demands of the consumers as intended. In this last stage, the user will test the system to see if it satisfies their needs as a business. The software will subsequently be released to production once this procedure is finished and it has passed.

9.1. Ethics Approval Application (only if required)

Not Required

9.2. Reflective Journals

10.0 **Supervision & Reflection Template**

Student Name	Conor Keller
Student Number	X18353923
Course	Bachelors of Computing (Cybersecurity)
Supervisor	Cristina Hava Muntean

11.0

12.0 **Month: October**

What?
This month was spent researching my project idea, for my chosen project the application will be a password vault accessible via PC, Tablet & Mobile, during this time I was tasked with submitting a project proposal video, within that video I explained what my project aims to tackle and how I want to approach this application, that was submitted to the Moodle page on October 8th
After this was submitted, I am awaiting to be allocated a mentor to assist me with the project. While waiting to be allocated a mentor I was tasked with drawing up a rough draft of my project proposal that was submitted on the 27th of October, this showcases my projects scope in its entirety but since it is only the first initial draft, I will have to change and adjust it over time.
As of October 26th, I was given my mentor known as Cristina Hava Muntean, she has organised for myself and her to have a one-on-one meeting secluded for week 8 of college to discuss the project.
So What?
I have successfully come up with an idea that was approved.
My next step is to have a meeting and discuss the scope and how I shall proceed with the given project.
The main challenged I faced was awaiting approval of the project, hopefully in the next journal I can supply a more detailed orientated reflection.
Now What?
I wait for my one-on-one meeting to discuss the scope with my mentor.

Student Signature	[illegible]

Student Name	Conor Keller
Student Number	X18353923
Course	Bachlors of computing (Cybersecuirty)
Supervisor	**Cristina Hava Muntean**

Month: November

What?
Met up with my mentor and discussed the project pitch idea.
Continued researching the idea via YouTube & GitHub.
Plans to begin starting the project skeleton on November 20th

So What?
Successfully managed to have the mentor meeting, but to be honest it was as useful as I thought it would be as getting an additional one on one meeting with my mentor seems to be impossible due to availability.
I am happy with the tutorials I have found online, and I am happy to continue on with making progress.
Skeleton progress has been layed out with more work done it will functionally be ready for my mid point in December 20th

Now What?
I want to successfully have the skeleton done and have the midpoint done a week in advanced.

Student Signature	[illegible]

Supervision & Reflection Template

Student Name	Conor Keller
Student Number	X18353923
Course	Bachlors of computing (Cybersecuirty)
Supervisor	**Cristina Hava Muntean**

Month: Decemeber

What?

Finally made a prototype of my project

Prepared myself for the mid point

So What? Successfully made an working prototype as a visual aid on how I want my project application to be like. Slowly put together my work and research together in preparation of the midpoint review
Now What? Complete the midpoint review and then research my progress and decided changes

Student Signature	[illegible]

Supervision & Reflection Template

Student Name	Conor
Student Number	X18353923
Course	Bachlors In computing (Cybersecurity)
Supervisor	**Cristina Hava Muntean**

Month: January

What? I completed my midpoint review at this of the project and have decided to stick with my chosen methods of developing the mid-point and have continued to clean up and user home page and login page thus far. I hope to have a meeting soon with my mentor once we receive the results so I can have a discussion on my struggles with the project thus far Once I am happy with the user registration I will move onto creating the first instance of my vault functionality

So What? From reflecting upon the midpoint progress on the project is at a steady pace as I have a clear path to follow and have happily cleaned up my user login/logout page for the web-based application, I am just awaiting results to decide upon a clearer path for my project . As the meeting will help me address the struggles I have been facing and to stress the urgency to myself as I have been taking this ordeal to blissfully.
Now What? See my mentors comments over the midpoint and decide a path from their on

Student Signature	[illegible]

Supervision & Reflection Template

Student Name	Conor
Student Number	X18353923
Course	Bachelors In computing (Cybersecurity)
Supervisor	**Cristina Hava Muntean**

Month: February

What? user registration has been complete with email confirmation and an option for two step verification via mobile number (Not fully working but the code is sound and works accordingly) The vault system has been created but function on getting the vaults to work like google autofill has been rough and will need to get this function working hopefully within the next week
Now What? March is the most important month for my project I aim to have all functionality complete and then hopefully by the end of the month I can attempt to make my application live to work out the kinks and flaws in the application. I gotta refocus my efforts and push ahead on the project as I have been ketting my personal life interfere with the progress I should be making

Student Signature	[illegible]

Supervision & Reflection Template

Student Name	Conor

Student Number	X18353923
Course	Bachelors In computing (Cybersecurity)
Supervisor	**Cristina Hava Muntean**

Month: February

What?

user registration has been complete with email confirmation and an option for two step verification via mobile number (Not fully working but the code is sound and works accordingly)

The vault system has been created but function on getting the vaults to work like google autofill has been rough and will need to get this function working hopefully within the next week

Now What?

March is the most important month for my project I aim to have all functionality complete and then hopefully by the end of the month I can attempt to make my application live to work out the kinks and flaws in the application.

I gotta refocus my efforts and push ahead on the project as I have been ketting my personal life interfere with the progress I should be making

Student Signature	[illegible]

Supervision & Reflection Template

Student Name	Conor
Student Number	X18353923

Course	Bachelors In computing (Cybersecurity)
Supervisor	**Cristina Hava Muntean**

Month: February

What?

user registration has been complete with email confirmation and an option for two step verification via mobile number (Not fully working but the code is sound and works accordingly)

The vault system has been created but function on getting the vaults to work like google autofill has been rough and will need to get this function working hopefully within the next week

Now What?

March is the most important month for my project I aim to have all functionality complete and then hopefully by the end of the month I can attempt to make my application live to work out the kinks and flaws in the application.

I gotta refocus my efforts and push ahead on the project as I have been ketting my personal life interfere with the progress I should be making

Student Signature	[illegible]

12.1. Other materials used

N\A