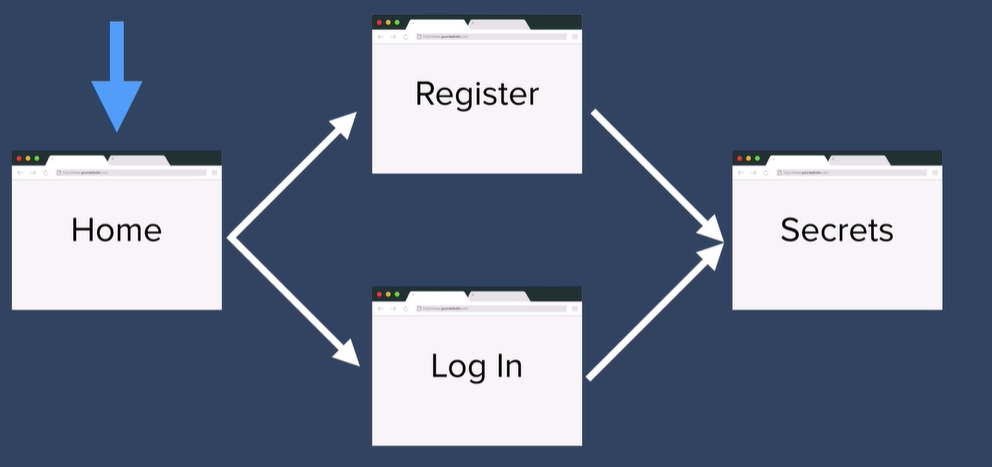
Day 68 Authentication with Flask



Figure out how to register, login and logout users with email and password. So they can access their own private profile pages.

Also, we're going to allow users to download a top-secret Flask Programming Cheat Sheet. But only when they have registered and signed up to our website.



Downloading Files

When the user accesses the **secrets.html** page, they should be able to download a secret file. The file is located in the starting project:

static > files > cheat\_sheet.pdf

In order to do this, we need to use a method from Flask called send\_from\_directory().

1. First go into the secrets.html page and make the anchor tag make a GET request to your server at the path /download

2. In the download route, use the documentation for send\_from\_directory() to download the cheat\_sheet.pdf file when the user clicks on the "Download Your File" button.

<https://flask.palletsprojects.com/en/1.1.x/api/#flask.send_from_directory>

Encryption and Hashing

<https://cryptii.com/>

<https://www.youtube.com/watch?v=G2_Q9FoD-oQ&ab_channel=Numberphile>

<https://www.youtube.com/watch?v=V4V2bpZlqx8&ab_channel=Numberphile>

<https://plaintextoffenders.com/>

<https://haveibeenpwned.com/Passwords>

<https://en.wikipedia.org/wiki/List_of_the_most_common_passwords>

<http://password-checker.online-domain-tools.com/>

Hashing Passwords using Werkzeug

At the moment, the user's password is stored in our database as plaintext:

1. Delete the previous unhashed entry in the database.

Let's secure their password by hashing it before we store it.

To do this, we'll use the Werkzeug helper function **generate\_password\_hash()**

2. Use the documentation here and see if you can figure out how to hash and salt the user's password:

<https://werkzeug.palletsprojects.com/en/1.0.x/utils/#module-werkzeug.security>

Aim to hash the password using ***pbkdf2:sha256***

And add a salt length of **8**.

This is what you should end up with:

Authenticating Users with Flask-Login

At the moment, if you simply navigate to /secrets you can see the secret page and the download link. There are no authentication barriers. How can we make sure that only registered/logged in users can see that page and download the file?

We'll need to secure certain routes in our server and make them only accessible if a user is authenticated.

To do this, most Flask developers will use the Flask\_Login package.

HARD CHALLENGE:

Use the [Flask\_Login documentation](https://flask-login.readthedocs.io/en/latest/" \t "_blank) to implement the /login route. The /secrets route should be secured so that it requires the user to be logged in.

This is what you're aiming for:

HINT 1: [You will need to configure your Flask app to use Flask\_Login](https://flask-login.readthedocs.io/en/latest/#configuring-your-application).

HINT 2: [You will need to create a user\_loader function](https://flask-login.readthedocs.io/en/latest/#how-it-works).

HINT 3: Make sure you [implement the UserMixin](https://flask-login.readthedocs.io/en/latest/#your-user-class) in your User class.

Note: A Mixin is simply a way to provide multiple inheritance to Python. This is how you add a Mixin:

class MyClass(MixinClassB, MixinClassA, BaseClass):

[Further Reading on Mixins](https://www.thedigitalcatonline.com/blog/2020/03/27/mixin-classes-in-python/)

HINT 4: [You can check the user's password using the check\_password\_hash function](https://werkzeug.palletsprojects.com/en/1.0.x/utils/#werkzeug.security.check_password_hash).

HINT 5: You need to find the user by the email they entered in the login form.

HINT 6: If the user has successfully logged in or registered, you need to use the login\_user() function to authenticate them.

HINT 7: Both the /secrets and /download route need to be [secured](https://flask-login.readthedocs.io/en/latest/#flask_login.login_required) so that only authenticated users can access them.

[SOLUTION](https://gist.github.com/angelabauer/f53574e00338a08e989c3c983506a8ba)

Flask Flash Messages

Sometimes, you will want to give the user some feedback on an action they took. e.g. Was there an issue with login in? Are they typing in the wrong password or does their email not exist? It would be a good user experience if, in these situations, we told them what was wrong, instead of just constantly redirecting them back to the login page.

The easiest way to do this is through Flask Flash messages. They are messages that get sent to the template to be rendered just once. And they disappear when the page is reloaded.

<https://flask.palletsprojects.com/en/1.1.x/patterns/flashing/>

1. Update the login route so that if the user's email doesn't exist in the database, you send them a Flash message to let them know and redirect them back to the login route. e.g.

HINT: A <p> tag in the login page will show up as red text.

2. Update the login route so that if the check\_password function returns False, you send a Flash message to the user when you redirect them back to the login page. e.g.

3. Update the /register route so that if the user enters an email that already exists in the database, you should redirect them to the login page and show a flash message to let them know they have already registered. e.g.

[SOLUTION](https://gist.github.com/angelabauer/b58cf1f98a9460eaf9a9adfd8d8b28e3)

Passing Authentication Status to Templates

When a user is logged in, the home page should not show the login/register buttons. And the navigation bar should not show Register or Login either.

e.g.

See if you can make some changes to the code in base.html and index.html so this happens.

Remember, as we learnt in previous lessons base.html is the layout template which all the pages inherit from.

<https://flask.palletsprojects.com/en/1.1.x/patterns/templateinheritance/>

[SOLUTION](https://gist.github.com/angelabauer/66f8d3a3532630998bee4f3a1fe3e517)