CSCB20
Introduction to Databases and Web Application

Week 9 - Sessions, Password Hashing, User authentication

Dr. Purva R Gawde

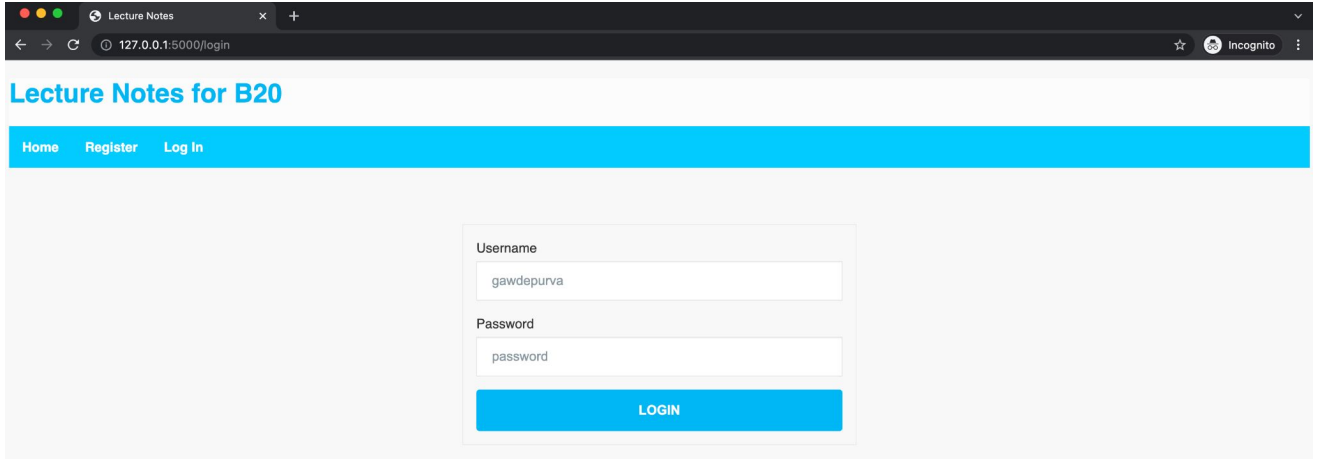
Topics

- Sessions
- User Authentication and Password Hashing
- Message Flashing

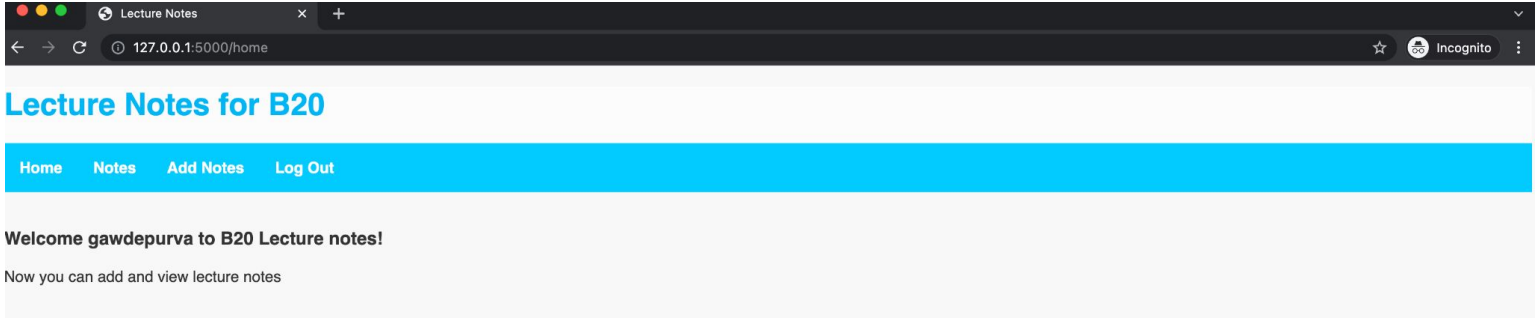
Sessions

Sessions

Login to your homepage (http://127.0.0.1:5000/login) with your username and password



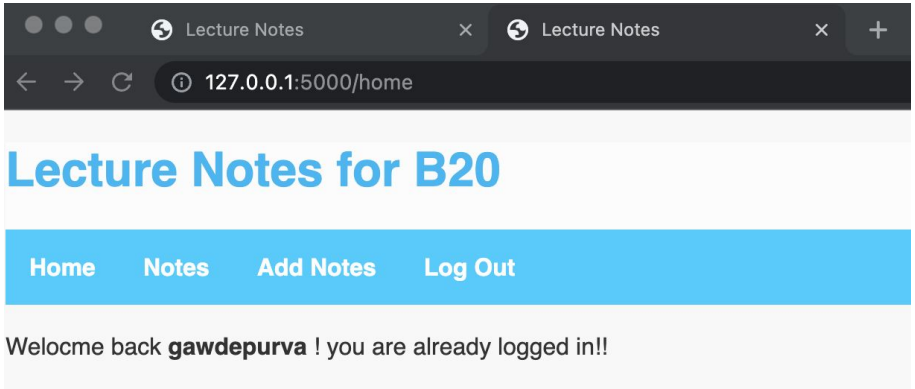
Once you log in, you get the welcome message



Sessions

- Next, you open up a new tab and visit the same URL again: (http://127.0.0.1:5000/login)
- What do you expect?
  - Login again

    OR
  - You are already logged in and welcome back message



Where, when and how does someone store this kind of data??

Introduction to Sessions

- What is a session?
  - A session stores information related to a user, across different requests, as they interact with a web app
- Why do we need sessions?
  - HTTP is a **stateless** protocol
- Where is session data stored?
  - On the **Server** temporarily
  - Example of data stored on server:
    - whether you are logged in or not
    - Items in shopping cart
    - Preferences for language
    - Dark mode vs light mode
- How do we protect our data stored in a session on the server?
  - Encryption - SECRET_KEY

Session

- Setting up a session

```
from flask import Flask, render_template, url_for, redirect, request, session
from datetime import datetime, timedelta
```

- Take care of the encryption
  - More on SECRET_KEY https://flask.palletsprojects.com/en/2.0.x/config/

```
app.config['SECRET_KEY'] = '84Br5667bb0b13ce0c676dfde280ba245'
```

- Session Object
  - Dictionary: contains key value pairs for session variables and associated values.
  - Example: to set a ‘username’ session variable, use the following statement:

```
session['name'] = admin
```

To release a session variable, use the pop() method.

```
session.pop('name', default = None)
```

Storing and Accessing Session Data

- Storing Data: Get the data you want to store in a session object from a class

```
session['name'] = request.form['Admin']
```

- Accessing Data stored in Session:
  - In app.py

```
if 'name' in session:
            return render_template('login_success.html')
```

  - In HTML Pages

```
{% if session.name %}
<h3> Welcome {{ session.name }} to B20 Lecture notes!
</h3>
  <p> Now you can add and view lecture notes </p>
  {% else %}
<p> Please login </p>
{% endif %}
```

Deleting Session Data

- Deleting Session

```
@app.route('/logout')
def logout():
    session.pop('name', default = None)
```

Session Life

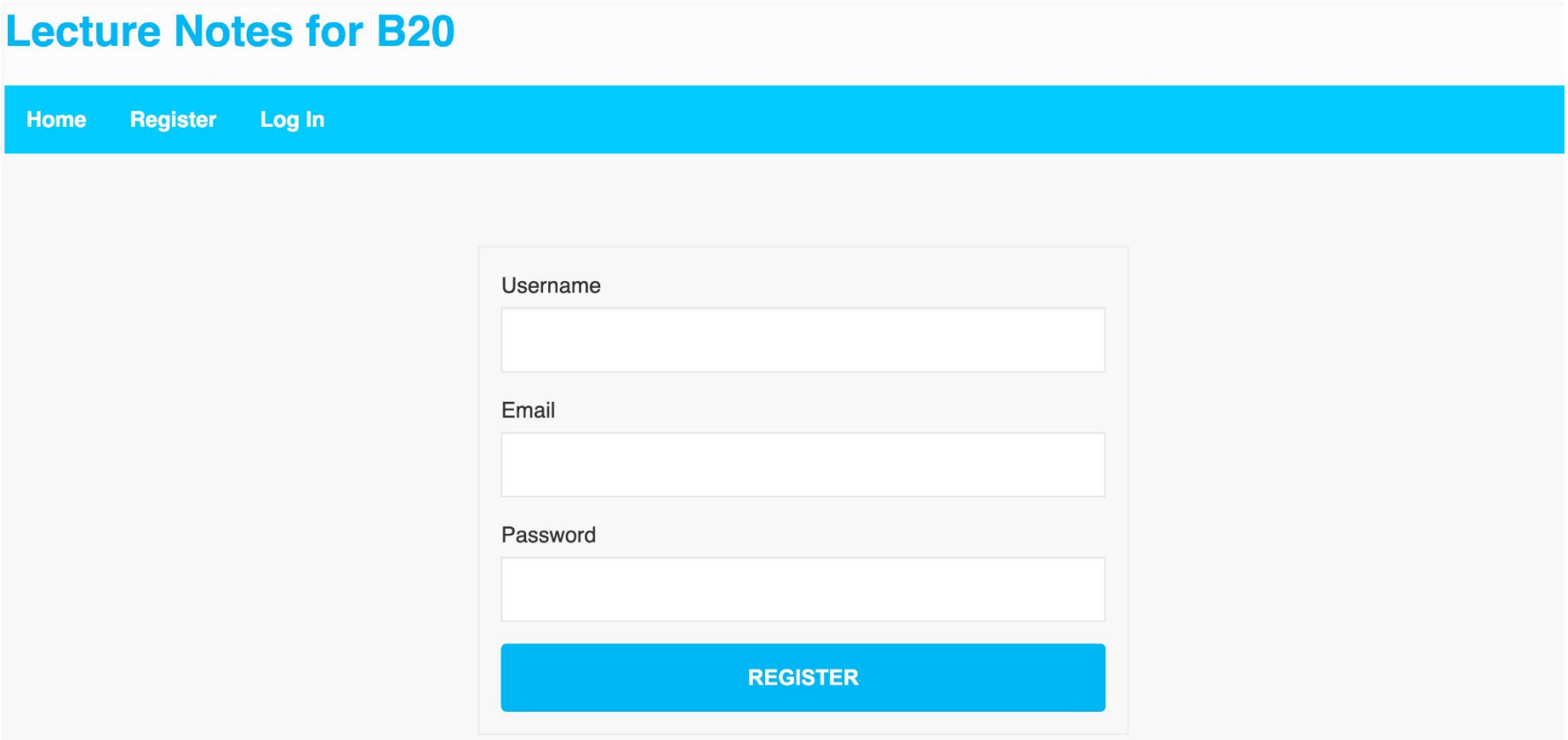
- Life of a Session: until the browser is closed
- We can increase the life of of a session using a config variable

```
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes = 10)
```

```
session['name'] = request.form['Admin']
session.permanent = True
```

User Login and Register

Registration Page



Registration => Adding users in DB

Steps:

1. Create a form
2. Take username, email and passwords from the form
3. When we hit submit, add that new user to the db

- Step 2 concerns:
  - Password visible to everyone with access to DB
- Solution:
  - Password hashing

Password Hashing

- Flask Extension - Bcrypt
  - Bcrypt - hashing function for password that is based on the **Blowfish cipher** and incorporates **salt** for protecting the application against any rainbow table attacks
- Installing Flask bcrypt module in python.

```
pip install flask-bcrypt
```

- Importing module and instantiating object of Bcrypt

```
from flask_bcrypt import Bcrypt
```

```
bcrypt = Bcrypt(app)
```

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

Authentication vs Authorization

- Authentication:
  - Act of validating that users are whom they claim to be.
- Authorization:
  - Process of giving the user permission to access a specific resource or function.
  - Gives client privilege/access control
- Bcrypt authenticates users by hashing their passwords and matching those passwords in order to authenticate users.

Bcrypt hashing

- Hashing a password named ‘password’

```
bcrypt.generate_password_hash('Password')
```

```
b'$2b$12$6D727ged9D2aXnpgtsasWOm7ZXXguIF0mkNaZJqyH73/O4qtRP2m2'
```

- Hashing a password named ‘password’

```
bcrypt.generate_password_hash('Password').decode('utf-8')
```

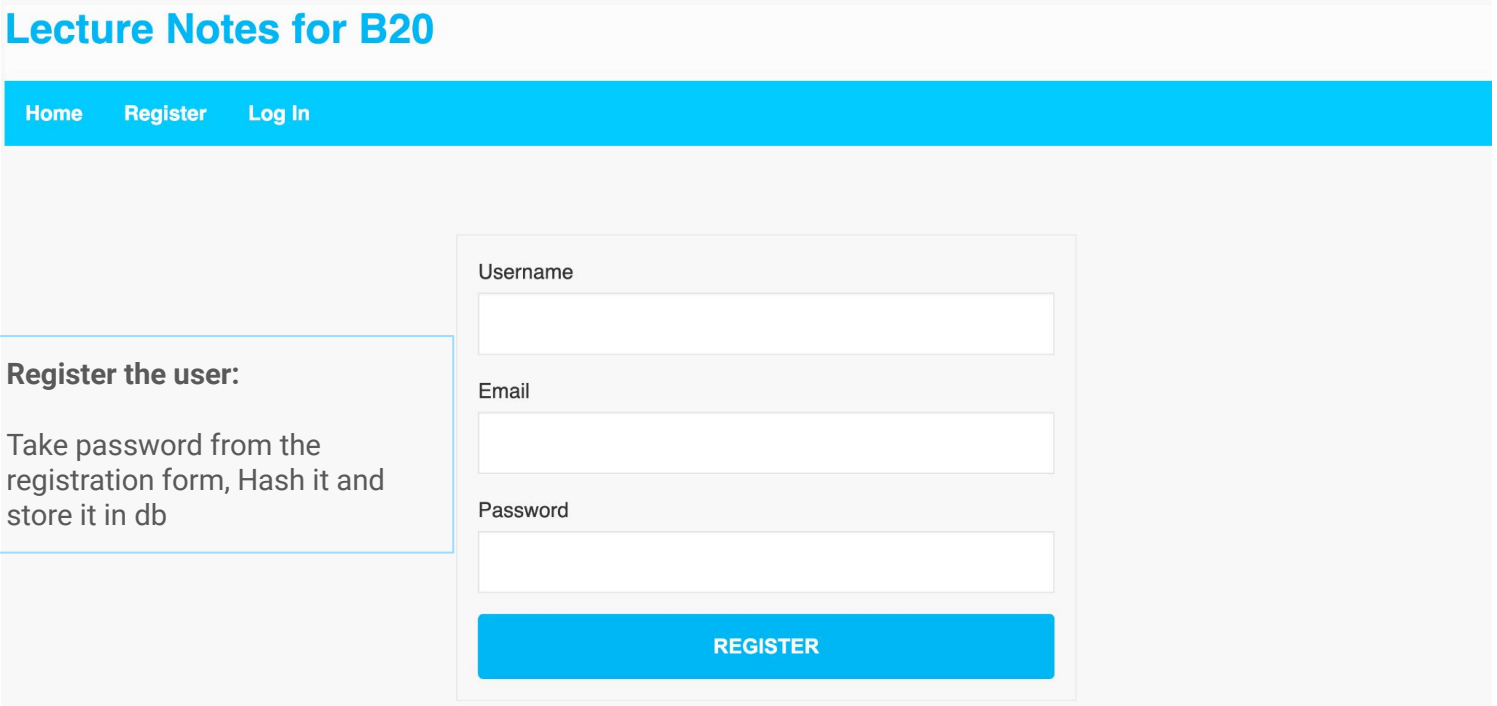
```
'$2b$12$JEOOVGt7aQoh2uJSSTsDEuqtMuPyE.q5Tu1xsSEaC1YGBYOfQDfZO'
```

- Each time, you run the hash function, you get different string.

  In order to match it, do following

```
hashed_pw = bcrypt.generate_password_hash('password').decode('utf-8');
bcrypt.check_password_hash(hashed_pw, 'password')
```

Registration Page

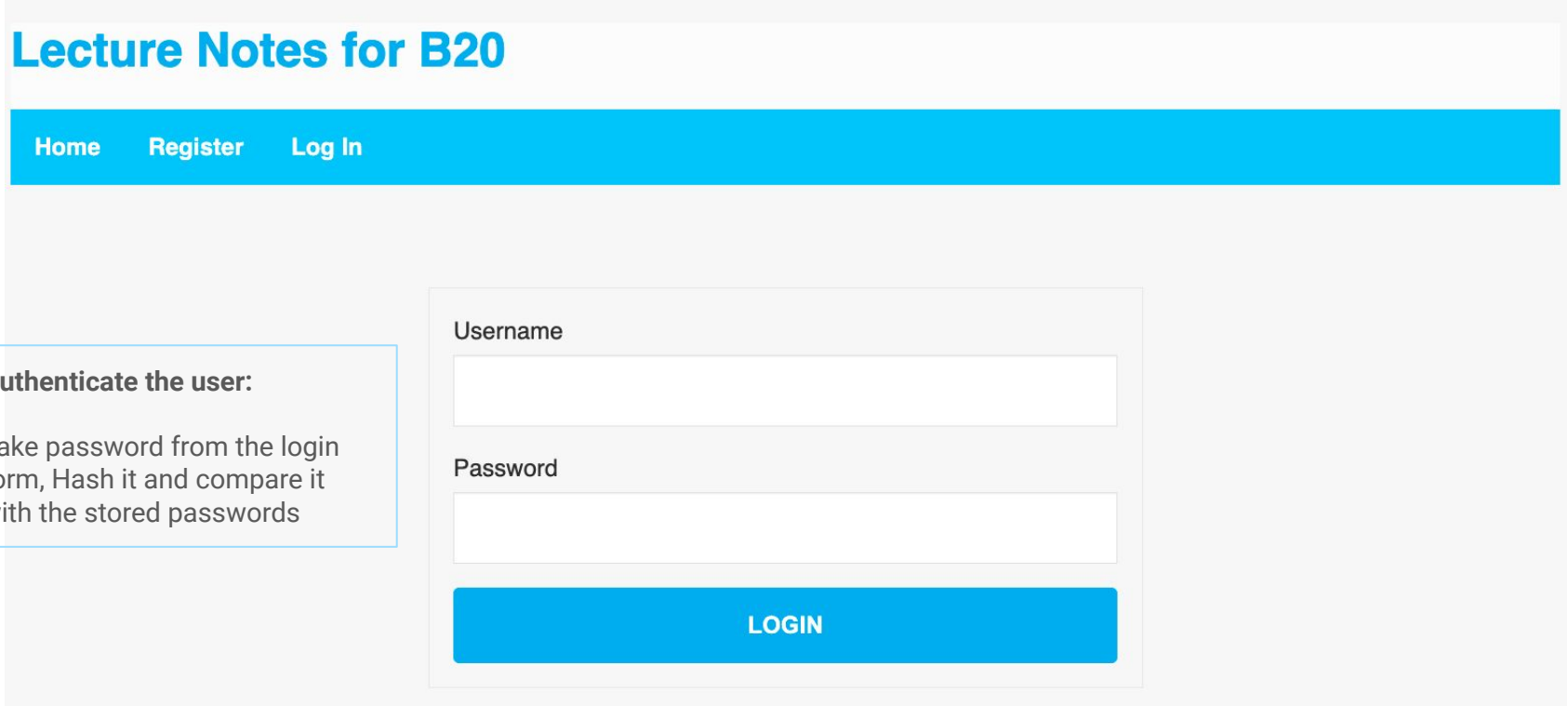


Register the user:

Take password from the registration form, Hash it and store it in db

```
hashed_password = bcrypt.generate_password_hash(request.form['Password']).decode('utf-8')
```

Login Page



Authenticate the user:

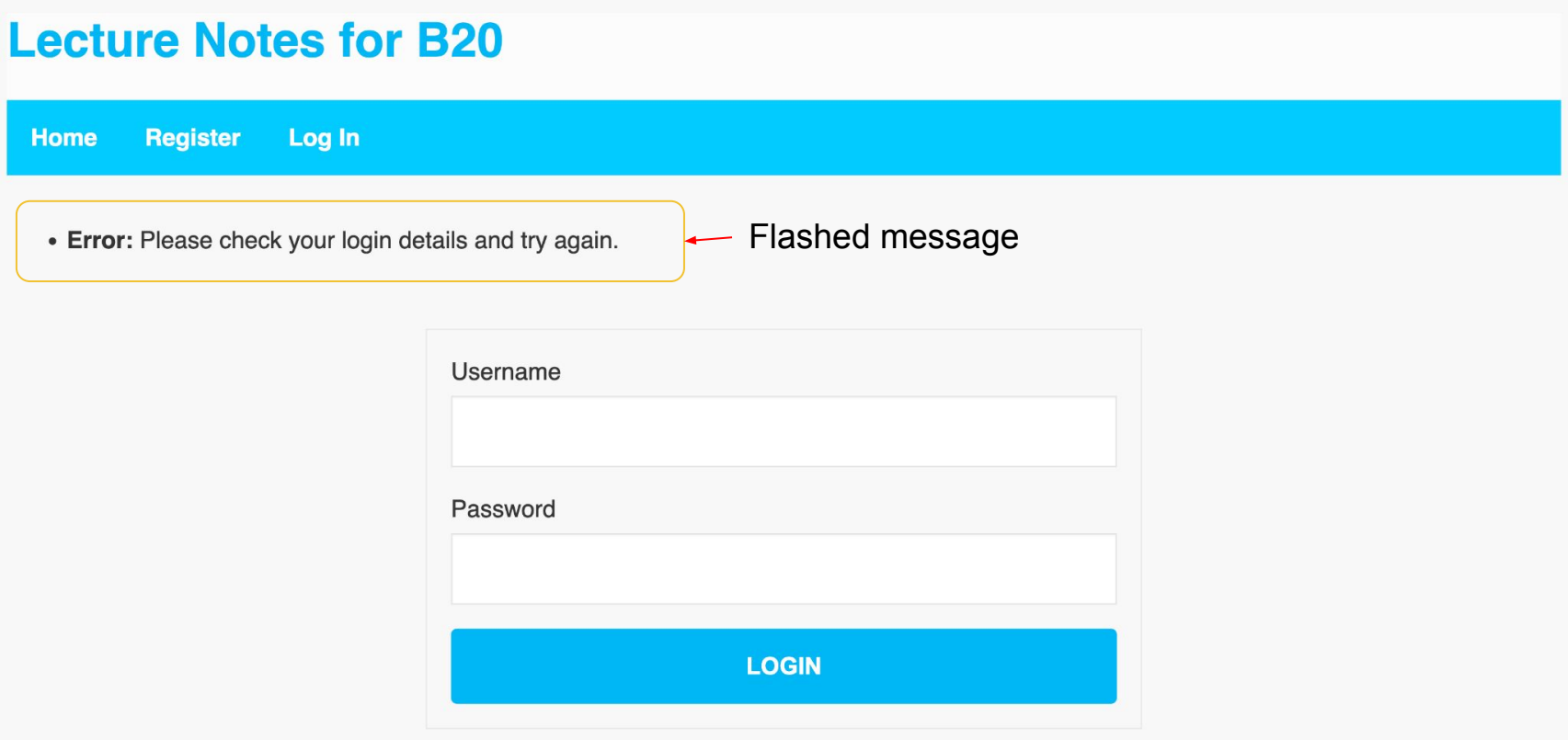
Take password from the login form, Hash it and compare it with the stored passwords

```
bcrypt.check_password_hash(request.form['Password'], password)
```

Message Flashing

https://flask.palletsprojects.com/en/2.0.x/patterns/flashing/

Message Flashing



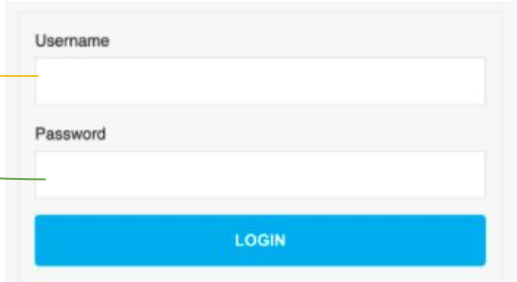
Flashing a message

- Import flash from flask

```
from flask import flash
```

- Check for users login details using Bcrypt

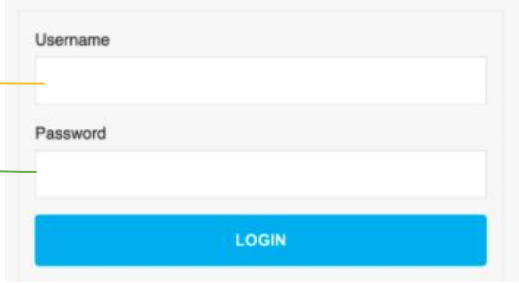
```
username = request.form['Username']
password = request.form['Password']
person = Person.query.filter_by(username = username).first()
if not person or not bcrypt.check_password_hash(person.password, password):
   flash('Please check your login details and try again.', 'error')
    return render_template('login.html')
```



Flashing a message

app.py

```
username = request.form['Username']
password = request.form['Password']
person = Person.query.filter_by(username = username).first()
if not person or not bcrypt.check_password_hash(person.password, password):
    flash('Please check your login details and try again.', 'error')
    return render_template('login.html')
```



login.html

```
{% extends "template.html" %}
{% block content %}
{% with messages = get_flashed_messages(with_categories = true) %}
{% if messages %}
   <ul class=flashes>
   {% for category, message in messages %}
     <li class="{{ category }}"><strong>Error: </strong>{{ message }}</li>
   {% endfor %}
   </ul>
 {% endif %}
{% endwith %}
```