This is an app that allows different users to upload images.
The application then watermarks this images with the user's identifier for copyright protection.

Installing packages

Copy the starter code for today in the folder for this course on your computer.

- Cd into the folder you just copied and create a virtual environment called `env`
- Activate the virtual environment (Windows: `env\Scripts\activate`, Linux: `source env/bin/activate`) and install the requirements (`pip install -r requirements.txt`)

Initial setup

- Create a new file called `db.py` and paste the following code into it:

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()
```

- Create a new file called `app.py` in the folder and paste the following code into it.

```
import datetime
from flask import Flask, render_template, redirect, url_for, flash, request,
session, jsonify
from flask_sqlalchemy import SQLAlchemy

import json
import os
from werkzeug.security import generate_password_hash, check_password_hash
from werkzeug.utils import secure_filename
from flask_wtf import FlaskForm
from PIL import Image
import qrcode
import shutil
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, Length, Email, EqualTo
from flask_login import LoginManager, login_user, logout_user, login_required,
UserMixin
```

```
from db import db
from models import User, Upload
from utils import *



app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'your_secret_key'



db.init_app(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
```

- Create a new file `forms.py` and paste the following code into it:

```
class LoginForm(FlaskForm):
    email = StringField('Email', validators=[InputRequired(), Email()])
    password = PasswordField('Password', validators=[InputRequired()])
    submit = SubmitField('Login')



class SignupForm(FlaskForm):
    email = StringField('Email', validators=[InputRequired(), Email()])
    password = PasswordField('Password', validators=[InputRequired(),
Length(min=4)])
    confirm_password = PasswordField('Confirm Password', validators=
[InputRequired(), EqualTo('password')])
    submit = SubmitField('Signup')
```

- Go back and modify `app.py`, add lines that import `LoginForm` and `SignupForm` from `forms.py`

- Add the following code to the end of the file (this is the code that actually starts the server, so it should always be at the end of the file)

```
if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

Functionalities

Every functionality will be handled by a separate function that will be decorated. The functionalities we need to provide are:

- sign up
- log in
- logout
- home
- upload
    - During upload, we have to also watermark the image that has been uploaded by the user.

Sign up

Create a new function called `signup` that takes no arguments.
On the line before the function, paste the following code (known as a decorator).
`@app.route('/signup', methods=['GET', 'POST'])`

This code says that this function will be run when we navigate to `/signup` and it will accept both `GET` and `POST` methods.

In the function body, create an instance of `SignupForm` called `form`.

Write an if statement that runs if the `validate_on_submit` method of the form returns `True`. In the if statement:

- Create a variable `hashed_password` and assign it the value `generate_password_hash(form.password.data, method='pbkdf2:sha256')`
- Create a new user and save it to the database, paste the following code:

```
new_user = User(email=form.email.data, password=hashed_password)
db.session.add(new_user)
```

```
db.session.commit()
```

- Display a success message using the following code: `flash('Signup successful. Please log in.', 'success')`
- Redirect to the login page: `return redirect(url_for('login'))`
- **NB**: all of this should be inside the body of the if statement

Out of the if statement, but in the `signup` function, put the following code: `return render_template('signup.html', form=form)`
This code returns displays the `signup.html` page, passing it the form.

Login

Write a function, `login` with url `'/login'`, that accepts both GET and POST methods.

In the function body, create an instance of `LoginForm` called `form`.

Write an if statement that runs if the `validate_on_submit` method of the form returns `True`. In the if statement:

- Get the user with the submitted email using the following code: `user = User.query.filter_by(email=form.email.data).first()`
- Write an if statement that checks if the user is not None and `check_password_hash(user.password, form.password.data)` returns True. In the if statement:
  - Log the user in: `login_user(user)`
  - Redirect to the home function: `return redirect(url_for('home'))`
- Write an `else` statement for the above if statement that displays an error message: `flash('Invalid email or password.', 'danger')`
- In the body of the `login.html` page and pass the form too

Log out

Create a new function, `logout`, that runs on `/logout` and accepts only GET requests
Add the following decorator to the top of the function that requires the user to be logged in: `@login_required`

Paste the following code into the function body

```
logout_user()
flash('You have been logged out.', 'info')
return redirect(url_for('login'))
```

Run the web app using `python app.py`.

Test test the different functionalities by navigating to their respective paths.

Home or index function

Write a function, `home` that runs on `'/'`, and requires the user to be logged in.

Paste the following code into the function's body

```
# get all the uploads from the DB
uploads = Upload.query.all()
return render_template('home.html', uploads=uploads)
```

Save the file and test the home function by going to `/`. Try logging out and going to the home function to see what happens.

Upload

Write a function called `upload_files` that runs on `'/upload'`, supports only `'POST'` and requires the user to be logged in.

- Create a variable, `current_user` and assign it the user currently in the session using: `load_user(session.get("_user_id"))`
- Create a variable `uploaded_files`, assign it the value `request.files.getlist('files')`
- Create a string called `current_user_wm` let its value be a string of the format `'id:<user's id>email:<user's email>'` (this string is not valid Python code, you are to write a code that creates a string with this format. **TIP**: use f-strings)
- Create an empty list called `file_paths`
- Loop through the `uploaded_files` list, storing each item in a variable called `uploaded_file`. In the loop:
  - Write an if statement that checks whether the `uploaded_file` has a truthy value. In the if statement
  - Get the current time and store it in a variable called now: `now = datetime.datetiem.utcnow()`
  - Format the `now` into a string: `now_string = now.strftime('%Y%m%d%H%M%S')`
  - Get a secure filename for the uploaded file and store it in a variable, filename: `filename = secure_filename(uploaded_file.filename)`
  - Split the `filename` created above using `'.'`, assign the first item in that list back to the `filename` variable. Assign the last item in that list to a variable called `'extension'`
  - Get the `file_path` by joining the `filename` to the `UPLOAD_FOLDER`: `file_path = os.path.join(UPLOAD_FOLDER, filename)`

- Save the uploaded file to the file*path:* `uploaded_file.save(file_path)`
- Create a variable `final_image_path` *where the watermarked image will be stored:*
`final_image_path = os.path.join(UPLOAD_FOLDER, f"{filename}{current*user.email}*
{now_string}.{extension}")* `- Embed the` current_user_wm `created above using the` embed_text_watermark` function. Here's how the function is called:
```python
embed_text_watermark(
input_file=file_path,
message=current_user_json.ljust(180, '-'),
output_file=final_image_path
)
```

```
        - Delete the original file: `os.remove(file_path)`
        - Add a new upload to the DB:



        - In the `upload_files`  function body, display a success message:
`flash("Upload successful", "success")`
        - Redirect to the home function

Run your code now and try to upload a file.
```