Review Document

Overview

This project is a personal portfolio website built using Flask, designed to showcase your work dynamically while also providing administrative functionality for managing content.

The following are the features I implemented:

1. Initial Project Setup

- Created a Flask web application to serve as my personal portfolio site.
- Set up basic routing to handle different pages such as the login page, home page and resume page.
- Added a basic structure to display project information dynamically on the home page using Flask.

2. Dynamic Project Display with Jinja

- Used Jinja templates in Flask to dynamically display project details on the home page.
- Looped through the project folders, reading relevant files (description, image, and PDF) and displaying them on the front end.
- Updated the project structure to include links, allowing users to click on project images and be redirected to external prototypes (e.g., Figma).

3. Admin Authentication System

- Implemented a simple authentication system with hardcoded admin credentials (username and password).
- Added admin-only sections of the website to provide more control over the content, such as the ability to add or delete projects directly from the front end.

4. File Addition and Deletion

- Added the functionality for an admin to upload new projects via the front end:
    - The admin can upload a project name, description, image, PDF, and link.
    - These files are stored in the Root Folder named ‘Projects’ and updated in the database for dynamic display.
- Developed the ability for the admin to delete a project:
    - The admin can select a project to delete by inputting the project ID, and the system will remove its corresponding files from both the database and the root folder.

- Normal users cannot see or access the project management (add/delete) functionality (form).

5. Database Integration with SQLAlchemy

- Integrated SQLAlchemy (with SQLite as the database engine) to store and display project data.
- Project details (name, description, image path, PDF path, and prototype link) are stored in the database, allowing efficient retrieval and updates.
- The project data is synchronized between the database and the file system (root folder) to keep everything in sync.

6. User Interface and Enhancements

Enhanced the portfolio's interface by:

- Making project images clickable and linking them to external prototypes when link is added by the admin (e.g., Figma prototype).
- Displaying project details dynamically based on what's in the database and root folder.
- Ensuring proper role-based access control by showing add/delete options only to the admin user.

Issues Encountered

1. It was time consuming to understand file paths and variables which led to broken links and missing images and pdfs.
2. Displaying images and pdf files and showing them dynamically (when admin uploads it from UI) was a challenge but I was able to fix it.
3. I think I've overcomplicated the code by including unnecessary loops and variables for deleting project files. The admin will provide a project ID, and using that ID, I need to delete the corresponding data from the database and local files.
4. I was not able to create a login in system where admin can set up his own username and password so for this version I hardcoded the values by setting them up in the main.py folder.
5. Moving from adding files manually into the VS Code to using a database was also hard to understand.
6. I am still figuring out on how to implement logout feature and how to put login button on top navigation bar.
7. Storing the resume option dynamically gave me quite a few errors. I think it is because it is navigation menu as a button.

How I plan on solving the issues

1. Create a button on top navigation bar and set it in an if condition such that only admin will be logout option (i.e., he should've been logged in already).
2. I will create another input form for resume so the admin can update the resume through the UI.

Where do I need help?

Currently, the login page isn't accessible from the home page because it's intentionally hidden. Instead, it can only be accessed by typing the URL directly (http://127.0.0.1:5000/login). However, I'm unsure if this is the best approach or if the login option should be visible on the home page.

Milestones for the next weeks

1. **Milestone 1: User Authentication Enhancement**
   Description: Improve the current user authentication system by implementing features such as password reset and user registration for non-admin users. This will ensure that all users can securely create accounts and recover their passwords while maintaining admin privileges.

2. **Milestone 4: Resume Update Interface**
   Description: Develop a dedicated form that allows the admin to upload and update their resume directly from the UI. This will streamline the process of keeping the resume current without needing to access the file system directly.

3. **Milestone 3: Logout Feature Implementation**
   Description: Implement a logout functionality that allows users to securely exit their sessions. This feature will improve user security and maintain the integrity of the admin controls.
4. **Milestone 4: User Feedback and Ratings System**
   Description: Introduce a feedback and ratings system where users can leave comments and rate projects. This will help in gathering valuable insights about user preferences and improving project visibility based on user engagement.
5. **Milestone 4: Update the styling for the form where admin adds/deletes Projects**
   Description: Create a css styling for the the form in index.html and make it more user friendly for the admin.

Self Reflection

- **How I Feel About Progress:**
  After making the recent changes, I'm pretty happy with how things are going. I've implemented some key features, like user login and project management(adding/deleting projects), and setting up the database up. Using Jinja to display the projects dynamically has made everything feel much smoother.

- **Finishing the Project:**
  I'm feeling confident that I can finish the project in the next few weeks. I think I can tackle the remaining milestones I mentioned.

- **Expectations vs. Reality:**
  - **What Was Easier:** The syntax clicked with me quickly, and it was fun to see dynamic content pop up on the site without too much hassle.
  - **What Was Tougher:** Setting up the SQLAlchemy and making it work smoothly with the app turned out to be more challenging than I expected. Figuring out how to structure the database and make everything flow right took a bit longer than I thought it would.

- **Aha Moments:**
  I had a big "light bulb" moment while working on the database. I realized how crucial it is to have a solid database structure from the beginning. It really makes a difference in how easily you can manage and pull data later on. This whole experience has shown me the value of planning ahead in software development—definitely a lesson learned!