Team Sharp Cat: Victoria Gao, Alvin Wu, Jonathan Lee, Madelyn Mao
SoftDev
P0: Da Art of Storytellin’ (Pt. 2)
2020-12-20

Scenario Two: web log hosting site

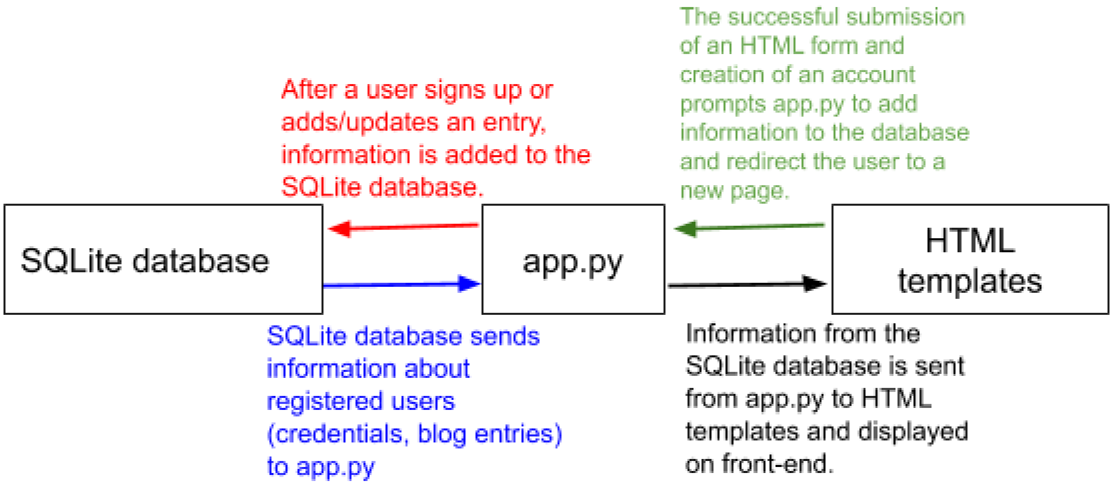
Components

- README.md:
    - Clearly visible at top: **<Project Name> by <Team Name>**
    - Roster with Roles
    - Description
    - Launch Codes:
        - How to clone/install
        - How to run
- SQLite database:
    - Table 1: Users
        - ID (non-negative auto-incrementing integer)
        - Username (unique field)
        - Password
        - Biography
    - Table 2: Posts
        - ID (of the post, non-negative auto-incrementing integer), used for sorting!
        - User ID field (all posts by registered users are in this table)
        - Text in entry
        - Date entry was last edited
- Flask app:
    - connects backend files (i.e. database) to frontend files (i.e. html templates). This Python script will redirect users to another webpage based on the buttons they click and their input in HTML forms.
- Templates:
    - index.html: home page that contains login/sign up buttons. Users must register/login to use the site.
    - signup.html: contains an HTML form that allows users to create a username and password for their account. **They will also enter a biography**. app.py will check if the usernames chosen by new users already exist in the Login table and verify if they fulfill other requirements (i.e. password length). Users will be redirected to response.html or error.html depending on whether their account was created successfully.
    - login.html: contains an HTML form that allows the user to enter username and password. When the user clicks the “Submit” button, app.py will check if the

credentials match an entry of the Login table in the SQLite database. Users will be redirected to response.html or error.html depending on whether they logged in successfully.

- error.html: If users fail to login or create an account successfully, they will be directed to this page and be told what went wrong. Below the error message, there is a “Sign Up” button that redirects the user to signup.html and a “Login” button that redirects the user to login.html
- response.html: If users login or create an account successfully, they will be directed to this page. This page will have a navigation bar with options like “Add Blog",“Update Blog”, “View Other Blogs” and “Logout”. If users click the “Create a new Blog” option, they will be redirected to addblog.html. If users click the “View other people’s Blog” option, they will be redirected to viewall.html. If users click the “Logout" option, they will be redirected to the home page index.html.
- addblog.html contains an HTML form with fields that allow the user to create a page name and add text.
- updateblog.html contains an HTML form with fields that allow the user to add text to an existing page. It has two buttons: “Save Changes” will change the content of the blog and “Discard Changes” will redirect users to the page they were viewing before clicking the “Edit” button.
- viewall.html displays a list of users who made blog posts in the recent week. If users click on a user, they will be redirected to a page with that user’s biography and blog entries, which is rendered by viewuser.html
- viewuser.html displays a list of posts made by a user with a certain id.
    - Blog post is a table with date and text
    - Sorted by Posts ID!!
- viewblog.html displays a blog entry
- logout.html: displays a message showing that the user logged out successfully. The user can go back to the home page index.html by clicking a button.

Component Map


The successful submission of an HTML form and creation of an account prompts app.py to add information to the database and redirect the user to a new page.
After a user signs up or adds/updates an entry, information is added to the SQLite database.
SQLite database
app.py
HTML templates
SQLite database sends information about registered users (credentials, blog entries) to app.py
Information from the SQLite database is sent from app.py to HTML templates and displayed on front-end.

Database Organization

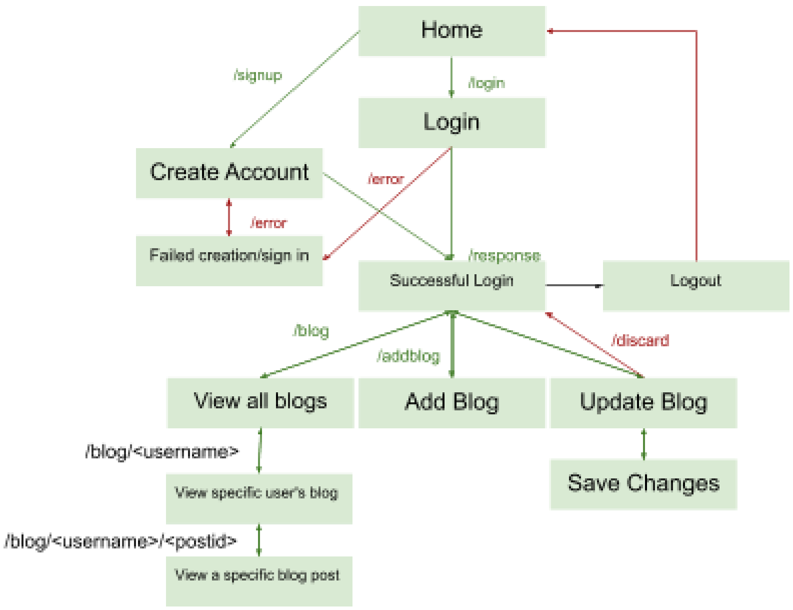
- Users table:
  - There will be a row for each user.
  - There will be 4 columns: ID, Username, Password, Biography

ID	Username	Password	Biography
0	blogger1	t0g2ka	blogger1 is currently a journalist working for the NY Times ...
<non-negative integer>	<username>	<password>	<biography>

- Posts table:
  - There will be a row for each blog entry.
  - There will 4 columns: ID, UserID, Text, Date

ID	UserID	Text	Date
0	0	Working on the blog project!!	2020-12-20-17:00:00
<non-negative integer>	<non-negative integer corresponding to a user in Users table>	<text in an entry>	<date entry was last updated, year-month-day-hour:minute:second>

Site Map for Frontend



Tasks

- Flask Python files (as necessary for “middleware” modules): Alvin Wu, Jonathan Lee
- SQLite Database: Everyone, creating/maintaining tables, mostly involved in the backend
- HTML templates/CSS: Madelyn Mao, Victoria Gao

Timeline

- 01/04/21-01/06/21: Basic functions of the website, login/logout, signup, add blogs, etc
- 01/07/21-01-08/21: Polish HTML files/add CSS and fix bugs