quality_assurance: Jun Tao Lei, Grace Mao, Sophie Nichol
P#00
SoftDev1 pd9
2019-10-27

Measure Once Cut Thrice

Naming Conventions:

- Blogs
    - Refers to the blog of a single user
    - In our case, the blog specifically refers to a **section** dedicated to the user
- Entries/Posts
    - Refers to the entries added to each user's blog
    - Entries may be confused with blogs depending on each team's nomenclature

The objective of this project is to design a basic blog that fulfills the following conditions:

- Ability to sign up by providing a unique username, a display name, and a user generated password
    - The display name will default to the username if the user did not choose one
    - The password will be stored in the database in the form of a hashed password
        - The salt will be generated by the uuid Python library
        - SHA-256 Hashing Algorithm used
- Ability to login with a username and a password
- Displaying the entries by the logged in user
- Simple Search Implementation:
    - Search by entry title
    - Use sqlite3's LIKE statement
- Adding new entries authored by the logged in user that feature the following:
    - Entry title
    - Author name
    - Content
    - Timestamp (Last Updated)
- Modifying entries authored by the logged in user with the following functionalities:
    - User functionality:
        - Edit the title in the selected entry

                - Title for each entry should be unique
            - Edit the content in the selected entry
        - System functionality:
            - Timestamp updates when the entry is updated
- Ability to view entries generated by other users:
    - Assumptions:
        - User must be logged in to view other users' contents
        - User cannot modify other users' contents
    - Procedure:
        - Navigate to the homepage that displays a list of display names
        - Each list item is a link to a page that displays a list of the selected user's entries
        - The list of the selected user's entries can display links to the entries.
        - Display entries by chronological order of **creation**
- Ability to change user settings
    - Ability to change password
    - Ability to change the display name
- Ability to logout

Since each page of a blog should have a similar design, there will be a base template with two broadly scoped blocks: header and footer. The body block should also be included in the base template, albeit remaining empty for the child(ren) to modify.

The following core forms should be included:

- Login form that features the following:
    - Username input field
    - Password input field
    - Login button to submit the form
    - Signup button to redirect to the signup page
- Signup form that features the following:
    - Username input field
    - Display name input field
    - Password input field
    - Password input field to confirm the correct password has been inputted
    - Submit button to submit the form
- Edit form (capable of adding or editing an entry) that features the following:

    - Title text field
    - Content text area
    - Save button to add or update the form
    - Delete button to remove the post being edited

The database should contain a table that contains a list of users and corresponding entry tables for each user. The corresponding entry tables should be named the same as the user's username or another applicable naming convention.

The storedsalt table should contain the following:

- Primary key that must be 0 in order to enforce a single row table
- Salt

The users table should contain the following:

- Primary key that auto-increments to be used as id for the user
- Username
- Hashed password
- Display name

The blogs table should contain the following:

- Primary key that auto-increments to be used as the blog id for the user's blog
- Userid as the foreign key
- Author
- Title
- Content
- Timestamp

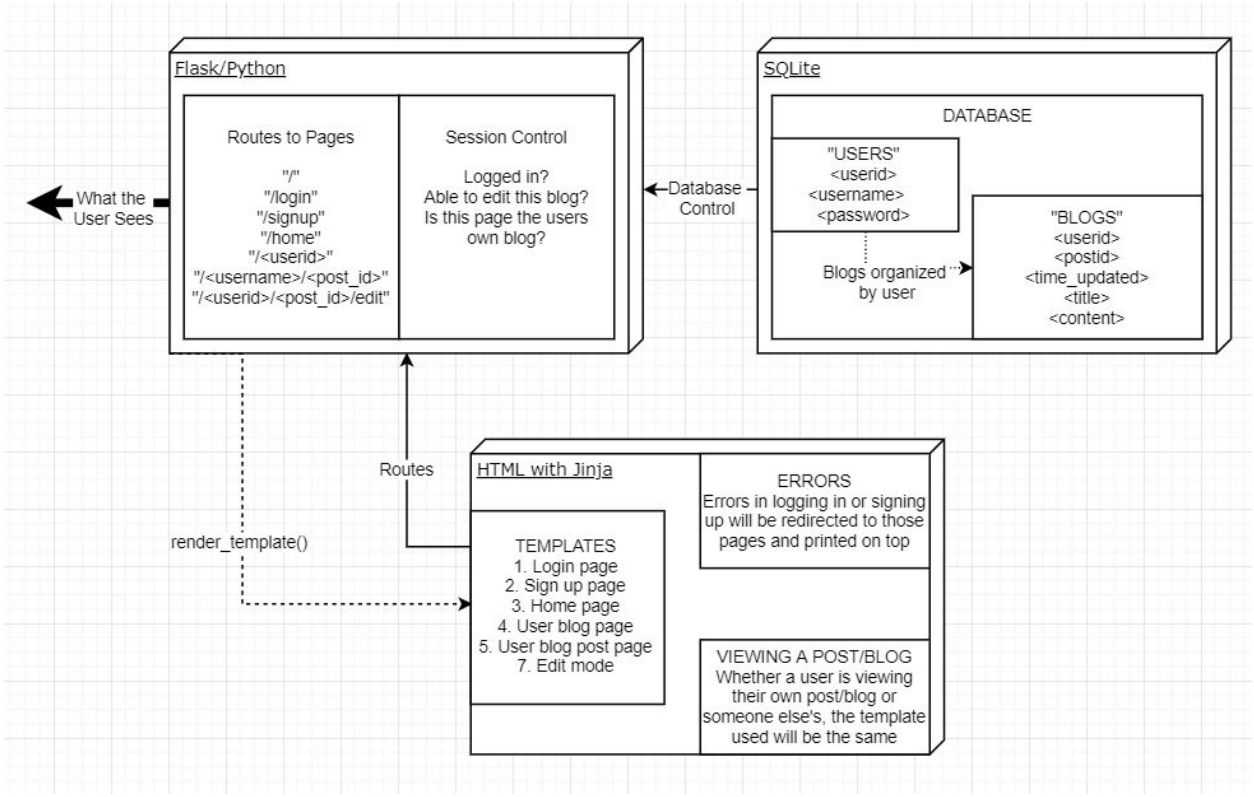
Assuming that a user visits this site and access the root route and that the user does not have a session, the user is prompted to the login page.

If the user does not have an account, the user can navigate to the signup page through a link in the login page. Otherwise, the user completes the required fields and submits the login form. If the operation is successful, the user will be redirected to the logged in homepage. Else, the user is redirected to root route that will show an error message. Likewise, the user will also be redirected to the signup page with an error message if the operation is not successful. Otherwise, the user will be redirected to the logged in homepage.

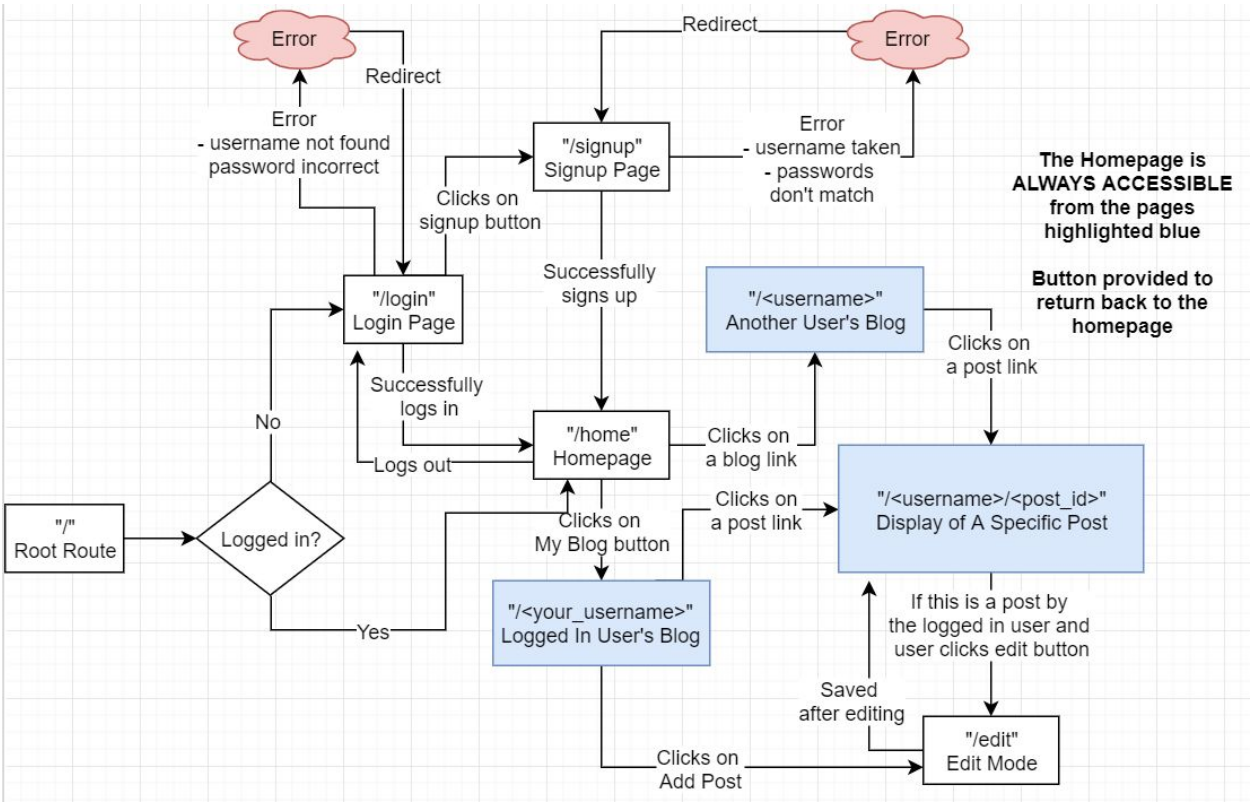
On the homepage, the user is able to navigate to a user's blog page as aforementioned. If the user has clicked on a user link, the user will be redirected to a page with all the entries made by the selected user. If the user clicks on an entry, the user will be redirected to that specific entry.

The "My Blog" button will redirect to the logged in user's own blog page. The logged in user's own blog will display a list of entries made by the user. If the user clicks on an entry, the user will be redirected to that specific entry. Under that page, the user has the ability to edit that entry. Otherwise, there should be an option to add a blog on the logged in user's main blog page.

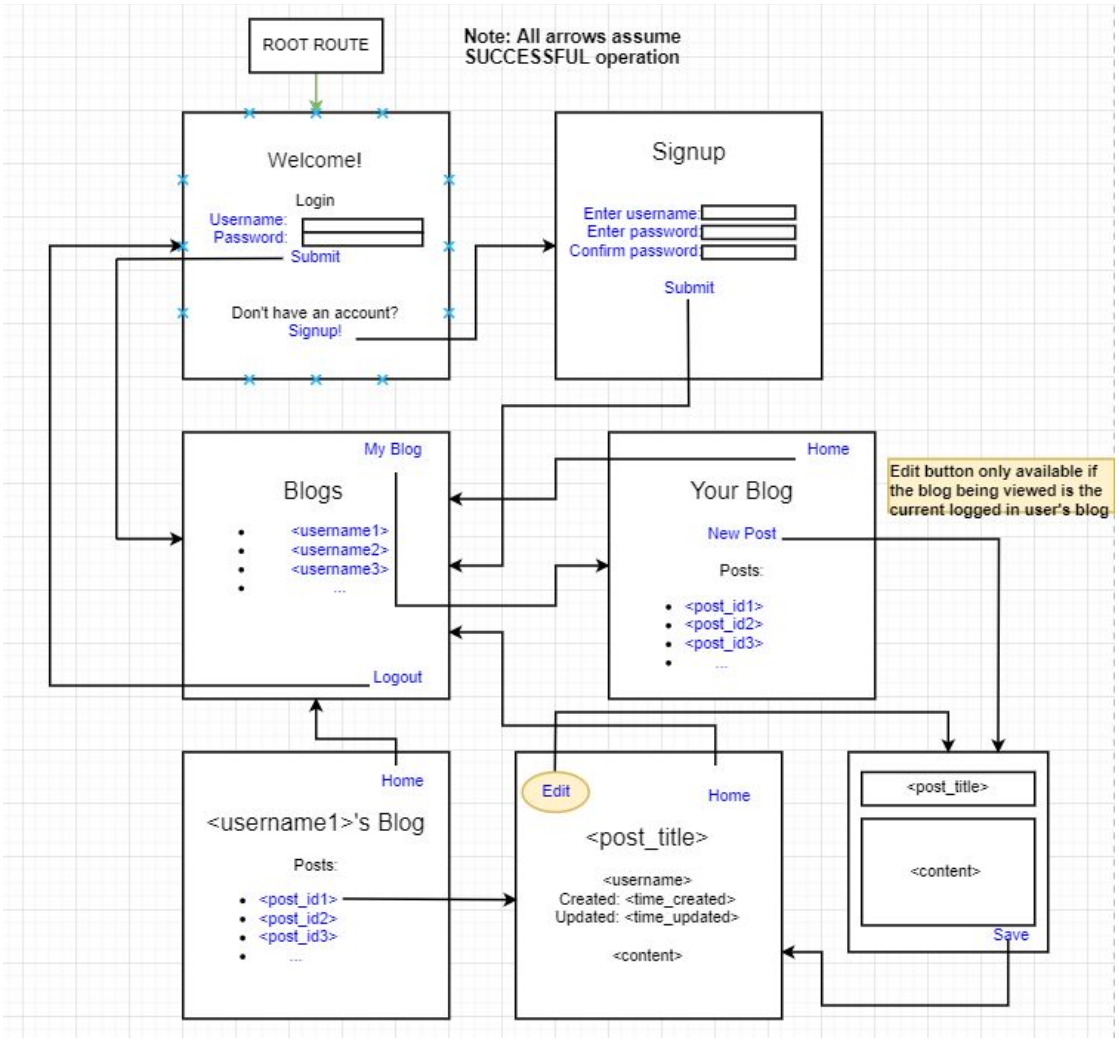
Component Map



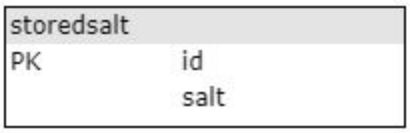
Route Map

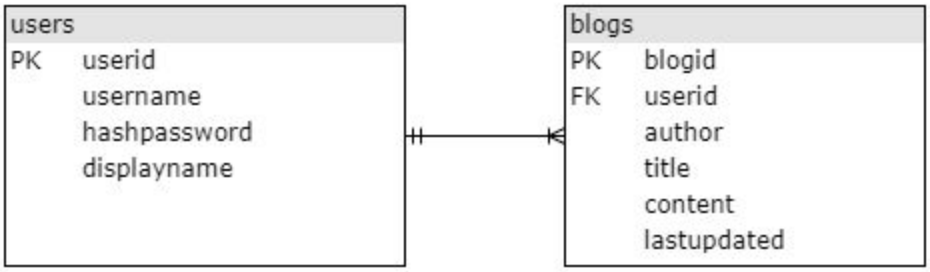

Error
Redirect
Redirect
Error
Error
- username not found
password incorrect
"/signup"
Signup Page
Error
- username taken
- passwords
don't match
The Homepage is
ALWAYS ACCESSIBLE
from the pages
highlighted blue
Clicks on
signup button
Successfully
signs up
"/login"
Login Page
"/<username>"
Another User's Blog
Button provided to
return back to the
homepage
Clicks on
a post link
Successfully
logs in
No
"/home"
Homepage
Clicks on
a blog link
Logs out
"/"
Root Route
Logged in?
Clicks on
My Blog button
Clicks on
a post link
"/<username>/<post_id>"
Display of A Specific Post
Yes
"/<your_username>"
Logged In User's Blog
If this is a post by
the logged in user and
user clicks edit button
Saved
after editing
"/edit"
Edit Mode
Clicks on
Add Post

Site Map



Database Layout Diagram





FLASK: Components

- Flask constructor is used to create the Flask app.
- g is used as a namespace object to store data during an application context.
- session is used to store data during a user's session.
- redirect is used to route the application.
- url_for is used as a method in redirecting to a different route.
- render_template is used a way to render templates in Flask.
- request is used as a method to receive data from client side.
- current_app is used as a proxy to access data associated with the current application.

PYTHON: Standard Lib

- search() is imported from re in order to use regular expressions.
- Number is imported from numbers to act as a type for most types of number.
- sha256() is imported from hashlib to perform the SHA-256 hash algorithmn.
- connect() is imported from sqlite3 to establish a SQLite3 Connection.

PYTHON: Self Created Modules

AUTH

- get_hash(password)
    - Return a hashed password
- auth(username, password)
    - Authenticate the user based on given credentials
- register(username, password, displayname)
    - Register the user if the user does not exist
- update_auth(username, currentpassword, newpassword)
    - Update the password given the current password is valid

DBCONN

- conn()
    - Setup a database connection and store it in the g object
- close()
    - Close a database connection that exist in the g object

DBFUNC

- header_types(tbl_name)

    - Return the column type(s) of a table
- insert(tbl_name, values)
    - Inserts a row into a table with the provided values
- get(tbl_name, column, conditional="")
    - Get certain data from a table based on the conditional(s).

EDIT

- create_post(userid, author, title, content)
    - Insert a row for a blog post into the blogs table
- update_post(blogid, blogcontent, blogtitle)
    - Update a row from the blogs table
- update_user(username, field, newvalue)
    - Update a certain attribute of a user in the users table
- delete_post(blogid)
    - Delete a row from the blogs table given the blog id

Roles:

- Grace Mao:
    - Templates
        - Make a template that will store and format the title, headers, content, and footers for each page of the blog to be displayed
    - Building Forms
        - Make the login and signup forms, as well as facilitate submission once a user is done editing a new post
        - Handle buttons in the templates (Home button on each page, submit buttons etc)
    - Facilitating Entry Addition/Update Using Built Database Operations
        - On a user's My Blog page, make a button "New Post", which will redirect them to a blank edit page (identified by the next available id) where they can put in their title and content
        - Make the "Edit" button on any existing post's page, provided that it is the logged in user's own post, and allow the user to update their table of posts in the database so that, once changes are submitted, the template will display the updated post
- Sophie Nichol:
    - Initialize Database

        - Make the Users table containing the username, id, and password for every user. For each user, make a <username> table containing a row for each post they make and storing the post id, title, contents, and timestamps
    - Connect Flask and SQLite3
        - Establish a database connection with Flask
        - Keep track of executions of SQLite commands and Flask actions
    - Database Operations Module (Insert to Table, Edit a Row, Create a Table)
        - Facilitate any changes that the user will request to the table, such as editing the table once a user updates a blog post, creating a new row when a user makes a new post, or creating a new table when a new user joins.
- Jun Tao Lei:
    - Project Manager
    - Route Functions
        - Renders the template for each page requested with routes pertaining to unique users generated using certain user attributes
        - Implement any other necessary functions or processes required for the route (i.e. call function(s) to fetch data from the database)
    - Login/Signup Module
        - Initialize a new session when the user logs in or signs up and remove it once they log out
        - Authenticates the user in a sensible manner