Members: Misha Kotlik, Zicheng Zhen, Seiji Yawata, Max Bertfield

Project Description:

Love subreddits but hate Reddit? Wish you could have your own group blog? Wish no more! <insert name here> is a self-hostable, customizable blog service featuring multi-user posting and (nested!) commenting, moderation, secure authentication, user profiles, and cross indexing by author, commentator, post, date, and tags. Everything you could wish for in a small and simple subreddit!

Legit Features:

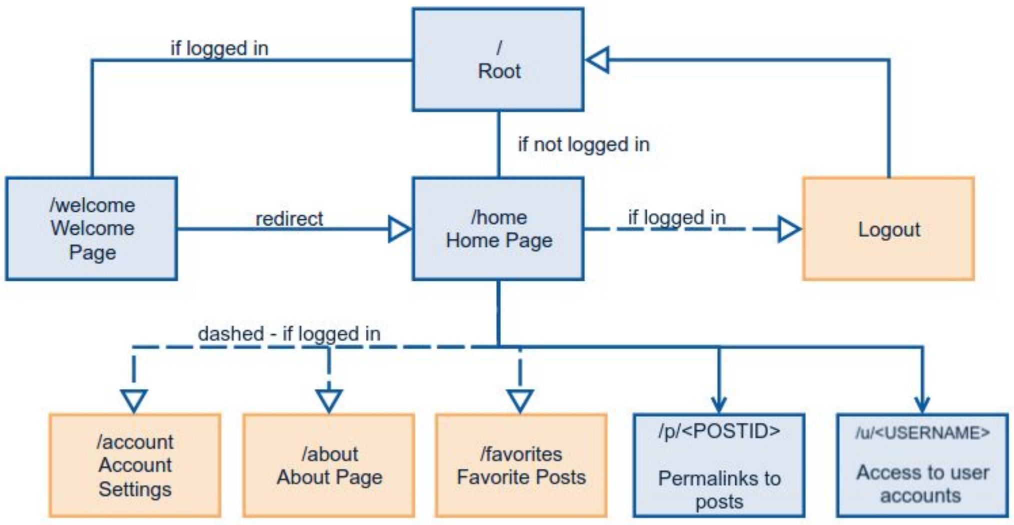
- One group blog - aka one thread but multiple users post to it (kind of like a subreddit or Stack Overflow)
- Nested commenting on posts - people can comment, reply to comments, etc.
- Upvoting of posts, displaying of posts by upvotes (over specified periods of times) vs date-posted
- User pages - show post/comment statistics for user, activity feed, + other info like Join Date, Bio, Most Popular Tags, etc.
- Saving posts - posts can be 'starred' or 'saved' for each user.
- Tags - posts can be tagged or put into categories. Also tagged by post date. Posts can be seen by tag.
- Authentication - users can/have to register with the website to post & comment. Use SQLite and Sessions for authentication service.
- Moderation - several levels of permissions can exist in blog, from owner/admin, to moderator, to regular user. Admin can appoint new moderators. Moderators can delete posts & comments, kick & ban users, promote posts.
- Customizable - admin can set user permissions, website styling, and turn features on/off

Database Schema

- Two databases: one for authentication and personal information, another for posts, comments, and the like.
    - Authentication Database:
        - Contains user credentials, such as usernames, emails and hashed passwords.

        - Schema: CREATE TABLE auth (userID TEXT, uname TEXT, email TEXT, pass TEXT)
- Blog Content Database
    - Posts table - fields include Post ID, Post Author ID, Post Date-Time, Post Content, Post Tags, Post Upvotes
        - Schema: CREATE TABLE posts (postID TEXT, authorID TEXT, time DATETIME, content TEXT, tagList TEXT, upvotes INTEGER)
    - Tags table - fields include Post IDs and Tags to allow for quick retrieval of posts with a specific tag.
        - Schema: CREATE TABLE tags (tag TEXT, postIDList TEXT)
    - Comments table - fields include Post ID, Comment ID, Parent Comment ID (default = None), Comment Content, Comment Author ID
        - Schema: CREATE TABLE comments (postID TEXT, commID TEXT, parentCommID TEXT, authorID TEXT, content TEXT)
    - User table - fields include User ID, User Name, User Post List, User Comment List, User Join-Date, User Last-Active-Date, User Permissions
        - Schema: CREATE TABLE users (userID TEXT, uname TEXT, joinDate DATETIME, activeDate DATETIME, perms TEXT, postIDList TEXT, commIDList TEXT)

<u>Site Map</u>



Python Modules:

- app.py
    - Main module of the service. Handles and processes requests through appropriate modules, including registration/login (auth.py), content posting/retrieval (content.py), user settings (users.py), moderation (users.py), server settings, sessions and cookies, and server initialization.
- auth.py (/utils)
    - Handles back-end involving login authentication and password hashing
    - Handles registration/login by reading from and writing to the auth database
- users.py (/utils)
    - Handles user info updating/retrieval by interacting with the content database. Used when a user registers, modifies their settings or account info, has their permissions changed by a moderator, posts content, or has their content viewed.
- content.py (/utils)
    - Handles posting/retrieval of posts and comments by interacting with the content database.

Additional Files:

- settings.ini
    - Stores blog-wide settings, such as view permissions (whether or not visitors need to be logged in to view posts/comments), links to css styling files, default post sorting on homepage, etc.
- static/avatars/<userID_pic>
    - Directory containing user's avatars, if they uploaded any, referenced by their userID
- static/*.css
    - Static directory for all CSS and Javascript
- templates/*.html
    - Directory with all HTML files

Breakdown of Tasks:

- Misha:
    - Deal with maintaining the authentication and personal info databases.

    - Look after python files making changes in the databases.
- Zicheng
    - Creation of posts and comments
    - Create nested commenting system
    - Create system for upvotes and tags in posts
- Seiji:
    - Manage login system in Flask app
    - Handle user-related actions (permission levels, creating new users)
- Max:
    - HTML templates for the Flask app
    - Create CSS templates and login/home interfaces