Mustafa Abdullah, William Li, Jiefeng Ou, Cody Wong
CJWM
Design Document
TARGET SHIP DATE: 2025-11-05

Scenario Δ: Creating a weblog hosting site

COMPONENT MAP:

Old:

__init__.py: Contains Flask and SQLite code

- Flask: Connects various components of the website together
  - Connects account info from SQLite databases to the end user
  - Has persistent login cookies
  - Connects html templates with data stored in databases
- SQLite: Stores info from the website
  - Account info, blog info, edit info (see database map)

New:

__init__.py : app setup and configuration

- Creates and configures flask app
- Sets up database connection path, key
- init/checks if sqlite database exists
- Registers blueprints so flask know where to find url mappings

models.py: database + data management

- Defines the components of the database (tables users, blogs, entries)
- Contains functions to initialize the database and also handle data
- Handles database connections, ensures consistent access to data

routes.py: webpage routing + logic

- Defines all the urls and what each one does
- Connects front-end with back-end
- Handles user actions
- Checks if users are logged in

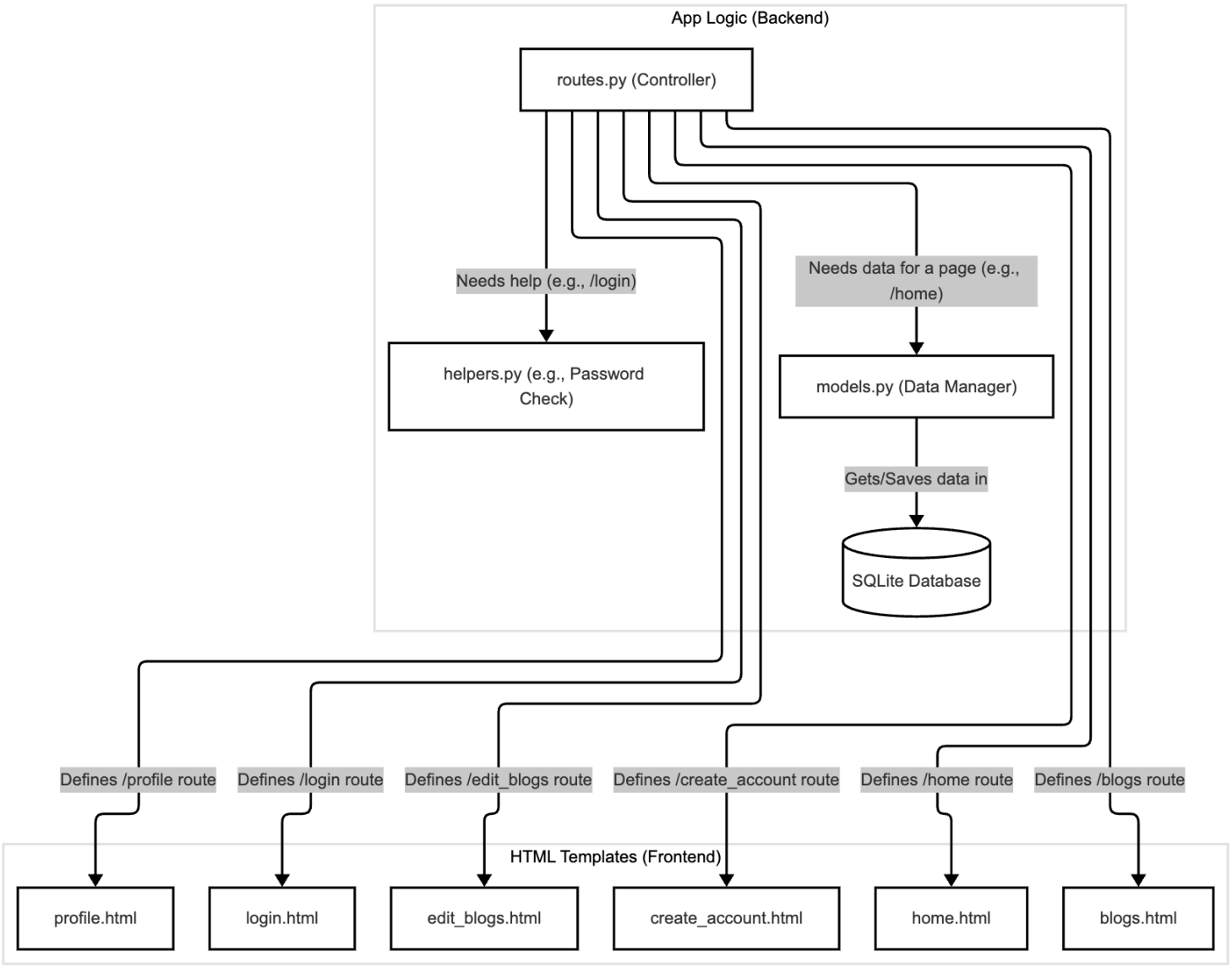
helpers.py

- Contains helper functions used across the app, such as password security handling
- (may not be necessary, will become evident with implementation)

HTML Templates:

- login.html: Checks login info to website, has persistent cookies that allow logins and deletes cookies when logging out
- create_account.html: Adds new login info to database

- profile.html: View users and blogs of those users
- home.html: Home page that allows users to find blogs
- blogs.html: Page that users see when viewing a blog
- edit_blogs.html: Page that users see when creating or editing a blog



SITE MAP:

Home Page: Accessible by a symbol in the top left corner (like how Google Docs or Github does it)

- Therefore, all parts of the site are able to get to the home page

Login Page: Accessible by all parts of the site except for the…

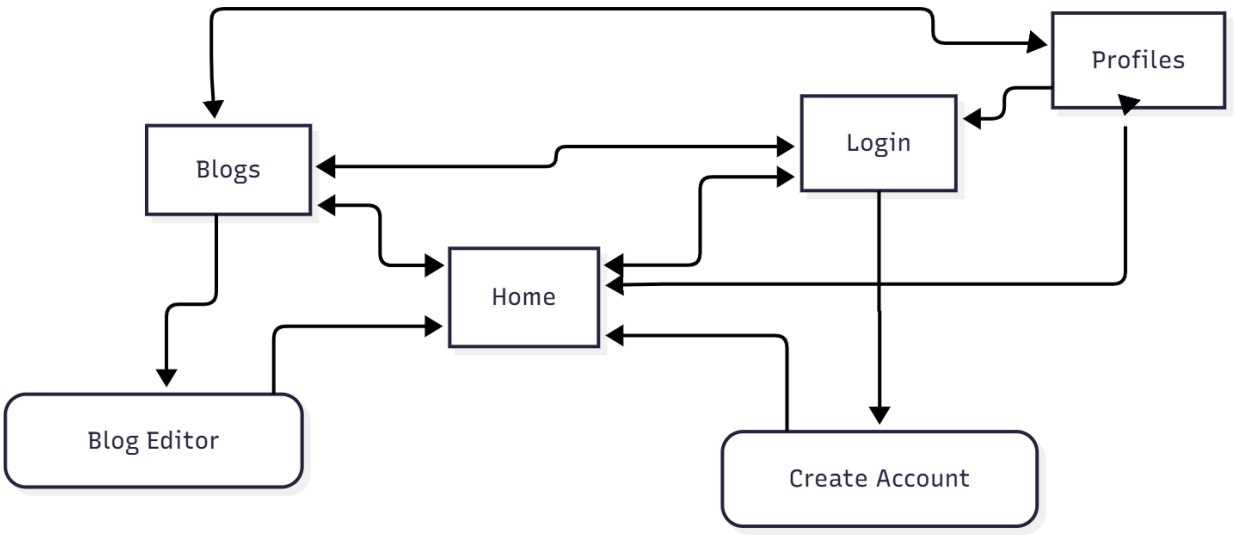
- …blog editing page (because logging in is required before you can edit)
- …account creation page (because there's no need to log in after having just created an account)

Create Account Page: Accessible through a redirect on login page, allows users to create accounts (with distinct usernames)

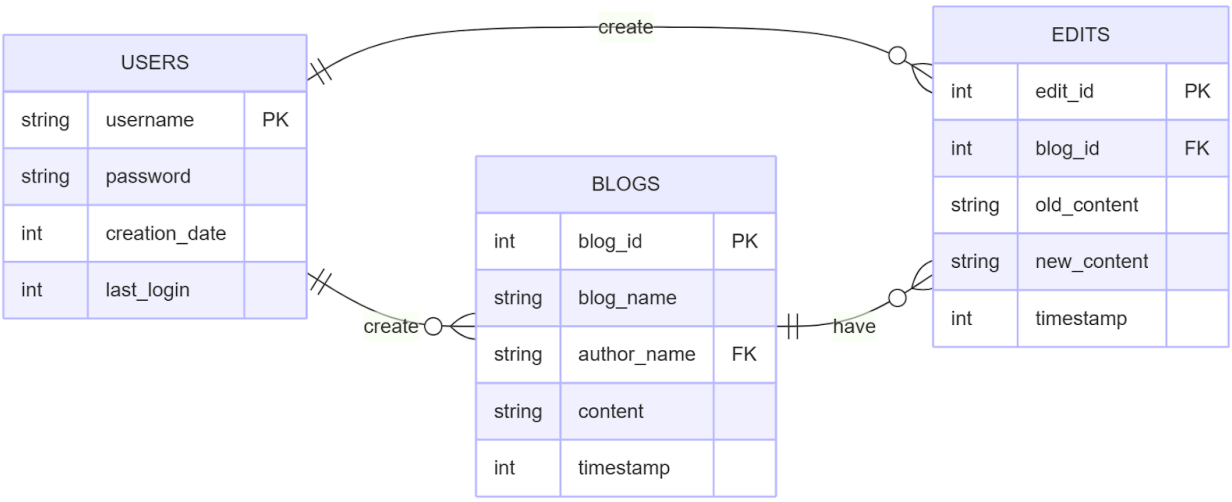
Profiles Page: Accessible through Home and Blogs Page, showcases information about user profiles (account creation date, last blog edited, etc.) and blogs edited by user

Blogs Page: Accessible through Home, Login, and Profiles pages. allows users to view past blogs created by their own user and other users. Blogs are sorted by the order in which they were created/last edited.

Blog Editor Page: Accessible through Blogs Page, allows users to edit their own past blog entries/add new entries to their current blog.



DATABASE MAP:



TASKS:

Create Account/Login/Logout pages: **Devo 1**

- Goal: implement user authentication, cookies in sessions
- __init__.py,login.html, create_account.html, home.html
- Tasks:
    1. Database Setup
        - Create users table in SQLITE that holds username, password, creation_date, and last_login
        - Make sure usernames are unique
    2. Routing with Flask
        - Handle login and logout forms, with logout clearing session data and redirecting to home
        - /login, /logout, /create_account
    3. Cookies
        - Redirect logged-in users appropriately
    4. HTML
        - Create login.html and create_account.html
        - Include "remember me" checkbox (maybe)?

Blog Viewing System: **Devo 2**

1. Database Setup
    - Creates blogs table in SQLITE3, with blog_id, blog_name, author, content, and timestamp
    - Sets up relation with blog author and user id
2. Routing with Flask
    - Handle displaying all blogs, sorted by date

- Handle displaying specific blog entries
- Handle searching for specific blogs based on title

3. HTML
   - Implement search bar at the top of home.html
   - Create "view blog" button

Blog Editing System: **Devo 3**

1. Database Setup
   - Connects blogs table to user sessions
   - Creates edits table, with edit_id, blog_id, old_content, new_content, and timestamp
2. Routing with Flask
   - Handle creating a new blog, editing blog if owned by user, removing a blog entry
3. Verify that a user owns a blog before allowing edits or removing things
   - Redirect unauthorized users to login page
4. HTML
   - Build a blog editor (title, text box, button to publish), with preview of past blog posts

Profile System: **Devo 4**

1. Handle profile page backend
   - Create profile route
   - Use users and blogs tables to display: username, account creation date, total blogs, last edit
2. Design profile.html
   - Show user info and list of blog titles
   - Include edit profile option
   - Add navigation links
3. Integrate
   - Link blog author names to their profile pages
   - Display profile links from home and blog pages of users
   - Check profile links to correct blogs