

## **P00: ~~Half-Quick~~ Move Slowly and Fix Things**

Design Doc by DECK, pd. 5

↳ Roster: Cindy Liu, David Lee, Ethan Cheung, Kalimul Kaif

### **PROJECT NAME: "LogDECK"**

A Flask application utilizing SQLite databases to store and display blogs. Readers will create or log into accounts to read others' blogs or create and edit their own.

**TARGET SHIP DATE: 2025-11-10**

---

## Program Components:

- ☐ `sqlite3` (backend data storage system)
  - ☐ `user_master_list` (stores relevant information for each user)
  - ☐ `blog_master_list` (stores relevant information for each blog)
  - ☐ `edit_history` (stores edit history of all blogs)
- ☐ `Python` (application layer)
  - ☐ `app.py` (runs Python)
- ☐ `Flask` (web server/delivery framework)
- ☐ `html` (frontend display)
  - ☐ `homepage.html` (displays user's blogs and other blogs)
  - ☐ `login.html` (checks entered login info against database)
  - ☐ `register.html` (adds new login info to database)
  - ☐ `profile.html` (displays user history)
  - ☐ `logout.html` (logs user out of session)
  - ☐ `create_blog.html` (enables user "blog" creation)
  - ☐ `edit_blog.html` (enables user "blog" edits)

### **Our MVP:**

A blogging platform with user login, viewable profiles, and a system for tracking contributions to blogs.

- SQLite (store user and blog data)
- Python (run scripts to serve HTML web pages)
- Flask (serve dynamic HTML pages)
- CSS (style the served HTML pages)

## Data Organization

- **user\_master\_list**
  - a master list of all users and related information (username, password, profile biographies, date of creation)
- **blog\_master\_list**
  - a master list of all the blogs with related information (name, creator, link, content, and date of the most recent edit)
- **edit\_history**
  - a master list of all edits made to any and all blogs (name of blog, blog creator, timestamp)

**user\_master\_list** is used by Flask in a Python script for:

- **login.html** (retrieve and compare user input login to system stored login info)
- **register.html** (add new user information to database)
- **profile.html** (display username, password, biography, and creation date)

**blog\_master\_list** is used for:

- **homepage.html** (display personal blogs and all other blogs available for viewing)
- **profile.html** (display the user's own blog information)
- **edit\_blog.html** (to access and update blog\_content & update last\_edited)
- **create\_blog.html** (to add new blog data to table)

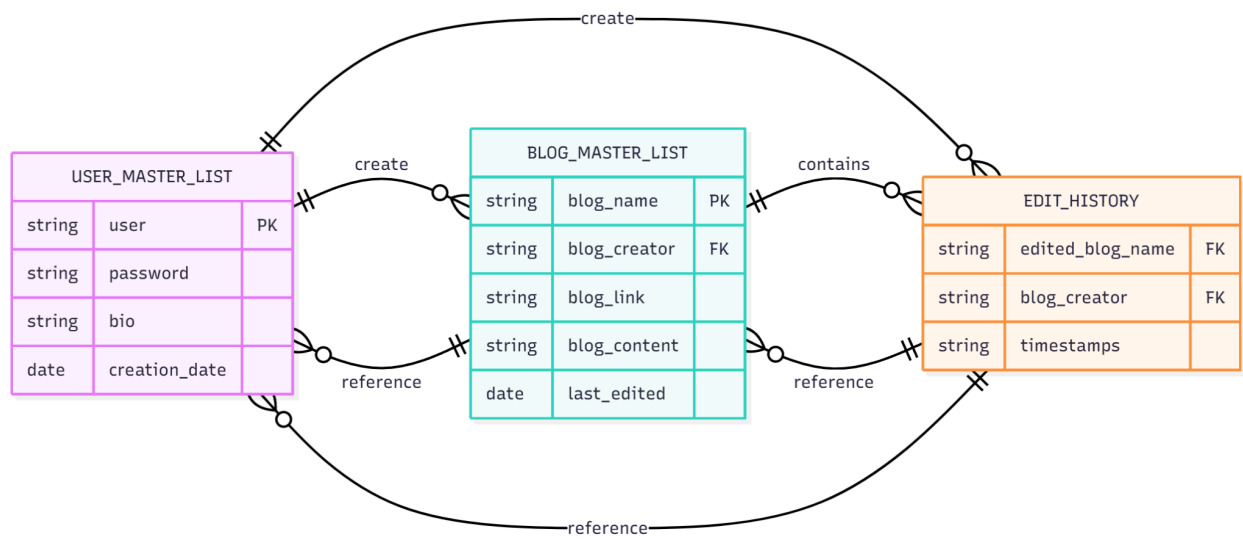
**edit\_history** is used for:

- **profile.html** (loop through all contributions, for matching username to append to a list of that user's edit history)
- **create\_blog.html** (append first contribution after form is submitted)
- **edit\_blog.html** (append new contributions after form is submitted)

All pages are dynamic in nature, because they each have user-specific outcomes (changes made to user state) and/or interact with at least one database for the addition/removal of displayed contents on each page.

---

# Component Map:



## Database Organization:

USER_MASTER_LIST			
TEXT	user	PK	unique identifier of blog user
TEXT	password		
TEXT	bio		user customization of profile
INTEGER	creation_date		only populated once during acc. creation

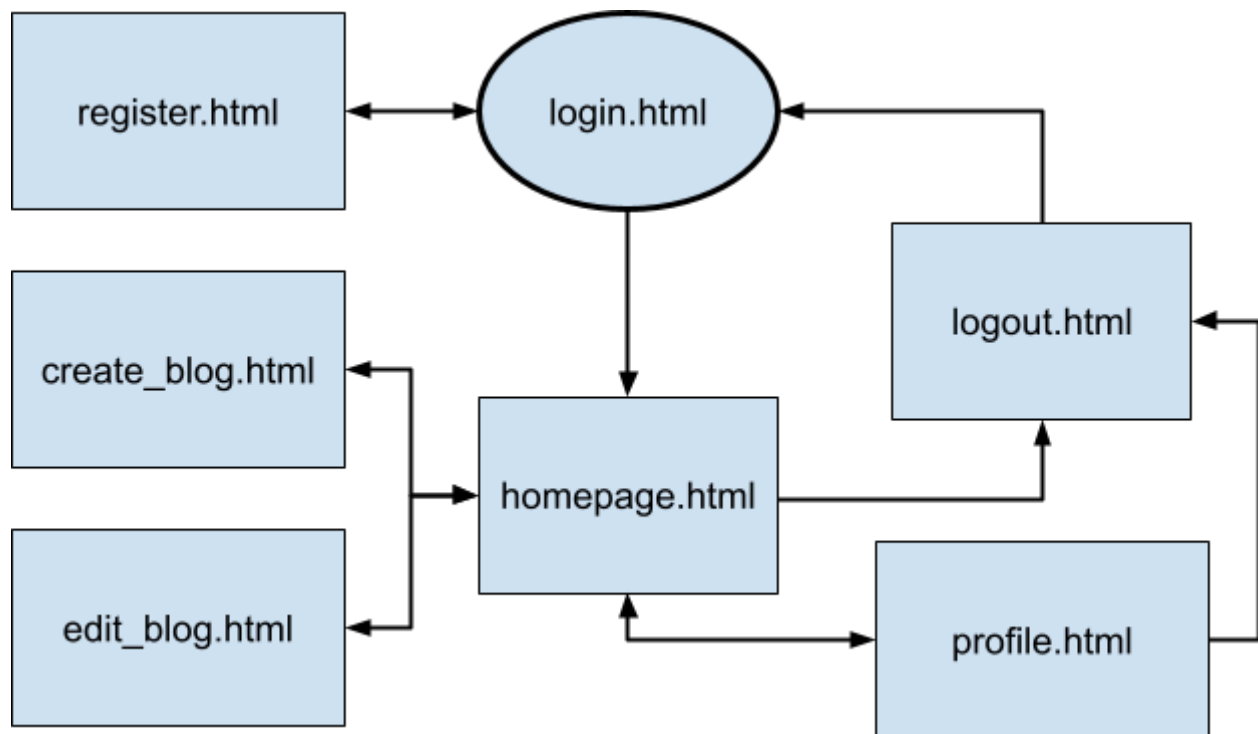
BLOG_MASTER_LIST			
TEXT	blog_name	PK	unique blog names
TEXT	blog_creator	FK	references users in user_master_list
TEXT	blog_link		
TEXT	blog_content		
INTEGER	last_edited		

## Database Organization (cont.):

EDIT_HISTORY			
TEXT	edited_blog_name	FK	reference blog names in blog_master_list
TEXT	blog_creator	FK	references users in user_master_list
TEXT	timestamps		

---

## Site Map:



## Breakdown of Tasks:

- ☐ create `__init__.py` (ethan)
- ☐ create csv files (david)
- ☐ setting up sqlite3 databases in `app.py` (david)

creating templates + corresponding functions in `app.py`:

- ☐ `homepage.html` (cindy)
- ☐ `login.html` (kalimul)
- ☐ `logout.html` (cindy)
- ☐ `register.html` (kalimul)
- ☐ `profile.html` (david)
- ☐ `create_blog.html` (ethan)
- ☐ `edit_blog.html` (ethan)