P02: Makers Makin' It, Act I

Design Doc by 2yellow2brown, pd. 5

↳ *Roster: Kalimul Kaif, Owen Zeng, Aoanul Hoque (PM), Yuhang Pan*

PROJECT NAME: 2Y2B TierList Creator
DESCRIPTION: Our website is a simplified tier list creator (inspired by tiermaker.com). Users cain sign up/login, create tier lists through an interactive editor, save them to their account, and view them later on a personal profile page. Other users can browse/search for tier lists and view creators' profiles to see the tier lists they've made.

TARGET SHIP DATE: 2026-01-16

Program Components:

1) Flask App (Python)
    a) **__init__.py** (main app + routes)
        i) Creates Flask app, config, and session handling
        ii) Renders templates and connects frontend to backend logic
        iii) Routes:
            (1) **/register** adds a new user to the **users** table (**data.py**). Checks if the username is unique, hashes the password, stores it in session, then redirects to **/dashboard**.
            (2) **/login** checks users (**data.py**), verifies password hash, stores session, redirects to **/dashboard**.
            (3) **/logout** clears session, redirects to **/login**.
            (4) **/dashboard** uses session + **data.py** to load the user's tier lists. Also includes a search bar. If a query is provided **(ex: ?q=...)**, it displays the filtered search results on the same dashboard page. Queries can be the title of the tier list or creator's username.

(5) **/create** inserts a new tier list into **tierlists** (owned by session user) and inserts default tiers into **tiers**, then redirects to **/edit/<list_id>**.

(6) **/edit/<list_id>** validates ownership, loads tiers/items (**data.py**), renders editor page. JS handles drag/drop and saving.

(7) **/view/<list_id>** loads tier list (**data.py**) and renders read-only view (public lists visible to all, private lists are only visible to the creator).

(8) **/profile/<username>** uses **data.py** to load a creator's public tier lists so other users can browse what they've made. It'll also show the username, and the # of tier lists they've made.

(9) **/delete/<list_id>** creator-only delete using **data.py**.

b) data.py

i) Connects to SQLite3 database and creates/maintains tables:

(1) **users:** stores user account information.

(2) **tierlists:** stores each tier list and who owns it.

(3) **tiers** (tier names + ordering per tier list).

(4) **items** (item label/image + which tier it's in + ordering).

2) Templates (HTML)

a) **/login** (username + password. Error message if login fails).

b) **/register** (username + password. Error message if username is taken).

c) **/dashboard** (shows most recent created tier lists + create button + search bar that filters results on the same page).

d) (**editor.html**) **/edit/<list_id>** (tier list editor UI (tiers + item pool + save button)) (JS handles drag/drop + saving).

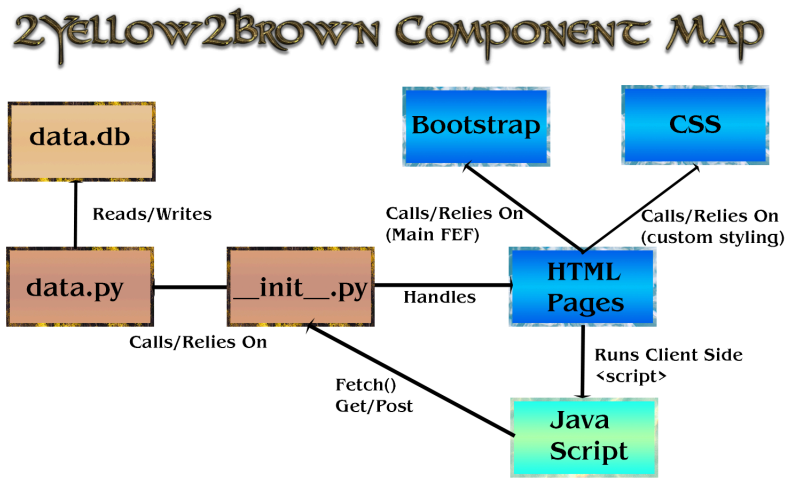
e) (**view.html**) **/view/<list_id>** (read-only tier list display for viewing).

f) (**profile.html**) **/profile/<username>** (shows username, # of tier lists, and their public tier lists).

g) **error.html** (invalid id's or permission errors).

3) Database (SQLite3) (stored in data.db)
    a) **users** table stores all usernames and password hashes for authentication.
    b) **tierlists** table stores tier list info (title/description/public/private) and the creator.
    c) **tiers** table stores the tier rows/ranking system the creator chooses (S/A/B/etc.) and their order for each tier list.
    d) **items** table stores draggable items (label + optional image url link), which tier they're placed in, and their order.
4) Frontend Framework
    a) **Bootstrap**: used for navbar, forms (login/registration), buttons, layout/grid, and tier list cards on dashboard/profile to keep styling consistent and responsive.
5) JavaScript
    a) Drag and drop items into tiers.
    b) Add/move items and update their page dynamically.
    c) Sends the updated tier list state to flask via POST so **data.py** can update the database.
6) RESTful APIs
    a) None
7) External CSS (if needed)

Component Map:



Database Organization

USERS

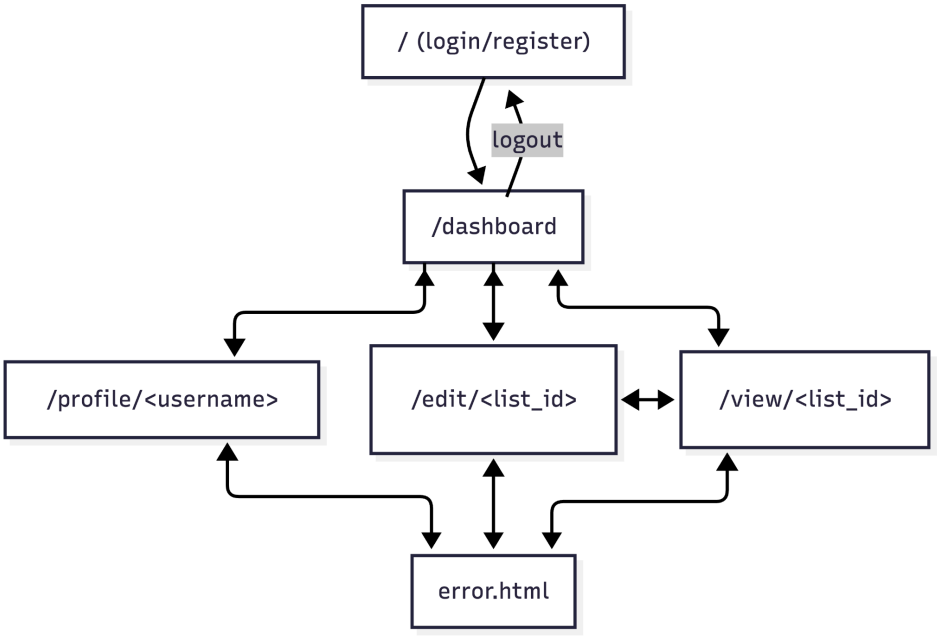
TEXT	name	PK	Unique
TEXT	password		For authentication

TIERLISTS			
INTEGER	id	PK	Auto-increment
TEXT	title		
TEXT	description		
BOOLEAN	is_public		Visible to other users?
INTEGER	creater_id	FK	Owner of tierlist

TIERS			
INTEGER	id	PK	Auto-increment
TEXT	name		S/A/B/etc.
BOOLEAN	is_public		Visible to other users?
INTEGER	tierlist_id	FK	Parent tierlist

ITEMS			
INTEGER	id	PK	Auto-increment
TEXT	name		Item name
TEXT	image		Optional image url
INTEGER	position		Order within tier
INTEGER	tier_id	FK	Parent tier

Site Map:



APIs

(None needed)

Front End Framework

Bootstrap CSS

Tasks & Assignments

DESIGN DOC WHO DOES WHAT

Task	**Devo(s)**
Program components & explanation	Aoanul Hoque
Component map	Aoanul Hoque
DB Organization	Yuhang Pan
Site map	Yuhang Pan
FEF section	Owen Zeng
Task Breakdown	Aoanul Hoque