Dogtree Eraser Lemons -- Karen Shekyan, Kevin Liu, Mahir Riki, Ian Jiang
Softdev
P01: ArRESTed Development
2022-12-05
time spent: 1.0 hrs
target ship date: 2022-12-23

APIs Used:
- Pokeapi
- Qrtag.net
- Superhero API
- Dad joke api

Framework: Bootstrap
- Greater clarity
- More familiar to the team
- Documentation easier to read
- Bootstrap features we're familiar with suit our needs (nav bars, search bars, cards, etc.)

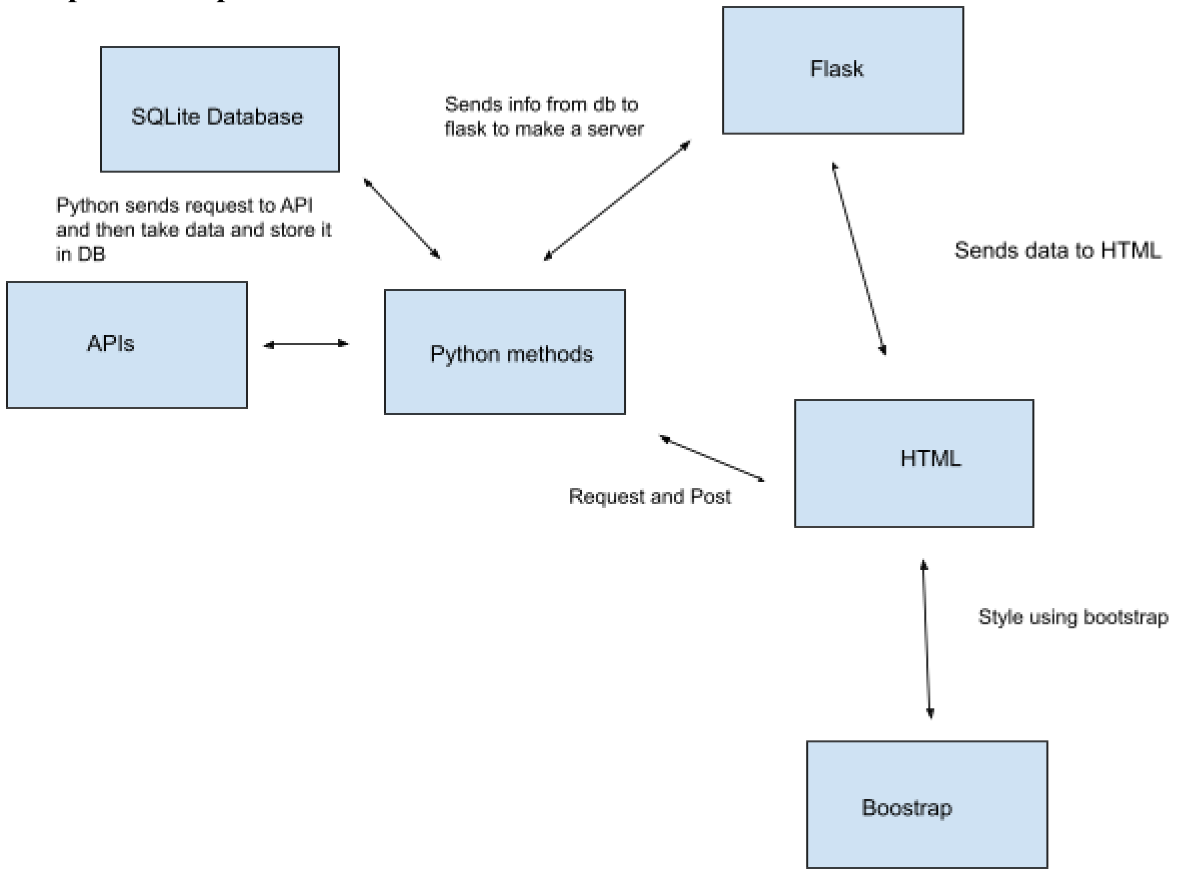
Components:
- A search bar that allows us to search up the different biographies of superheroes and pokemons.
- User accounts
- A route that allows users to sign up
- A route that allows users to log in
- A route that allows users to get information about Superheroes and Pokemon (search page)
    - An image of the hero/pokemon will be displayed, along with their stats and other info (pokedex, biography)
- A route that allows users to get a dad joke told between a random pokemon and superhero
    - Jokes can be favorited, which causes the joke to be stored in the database along with the characters who told it
- SQLite database that stores:
    - Usernames, passwords, and favorited jokes from the dad joke API (done with joke IDs)
    - Joke ID, joke content, pokemon, hero. This table is created for each user and contains information about each favorited joke
- A landing page with a button which will send you to a log-in route or sign-up route. The landing page is accessible from any route.
- A home page for the user when they are logged in. It will display your username to indicate that this user is logged in right now. It will allow a user to log out from any route.
- Error page for not found characters and profile
- Templates for:

    - Landing page
    - Home page
    - Search page
    - A user profile page
    - Three templates for pokemon data, superhero data, and a joke
        - Pokemon and hero templates are used to display their information on the search page
        - Joke template used for random joke

Connections between components:

- Navbar with search bar and buttons allowing users to go from any page to any page.
- User accounts allow users to log in and log out as well as sign up.
- A SQLite database that stores:
    - User login information and saved jokes (table 1)
    - Joke IDs, content, pokemon, and hero (table 2)
- The user profile will take information such as saved jokes from the database and display it on a per user basis.
- A landing page which allows you to sign up or login, allowing you to access the rest of our website.
    - The landing page is the home page for logged-in users
- A home page for the logged-in user to go to any other components of our website.
- Templates allow us to display information easier.

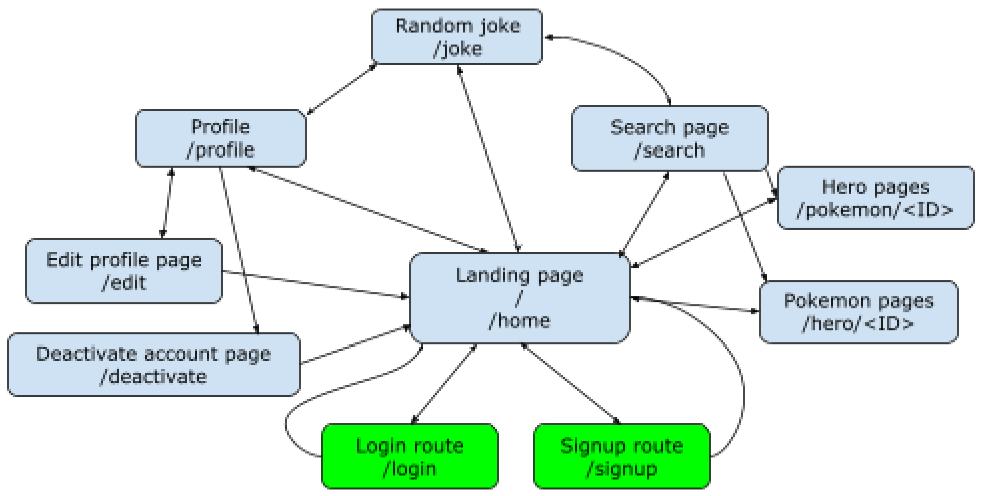
Component Map:


SQLite Database
Sends info from db to flask to make a server
Flask
Python sends request to API and then take data and store it in DB
Sends data to HTML
APIs
Python methods
HTML
Request and Post
Style using bootstrap
Boostrap

Database Organization:

- 2 tables:
    - User info
      create table user_info(username text, password text, favorite_joke_IDs text)
      This table will be used when logging a user in, to verify that their username matches their password
        - username is unique
        - user_id helps keep track of information we might need like total profiles
        - favorite_joke_IDs contains a list of favorite joke IDs (comma-separated), which can be appended to
    - Pages
      create table jokes(joke_id int, content text,);
        - Joke_id is our primary unique identifier
        - The content is the joke formatted in a way, lets say "Why did the chicken cross the road$%?Why?$%To get to the other side." Which we can split on $% and format.

Site Map:



Key:

- The format for each box is the title of the page, followed by its route
- Two-way directional arrows mean that two pages are accessible from one another
- One-way arrows mean that one web page redirects to another
- Blue boxes mean that this route is accessible via GET request
- Green boxes mean that this route is accessible via POST request
- **Important thing to keep in mind:** Every GET-accessible page has a link back to the root of the webpage (there is a logo in the upper-left corner than guides the user back to the start)

Description of each page:

- / — If you're logged in, then this is the home page. If you're not logged in, then this is the landing page. It contains a logo and the login and signup forms.
- /home — The home page contains buttons where you can visit every other page from.

- /login — The login route. This is the route that the login form on the landing page sends the information to
- /signup — The signup route. This is the route that the signup form on the landing page sends the information to
- /profile — Displays the content of the user profile. Mainly used to see the jokes that the user has favorited.
- /edit — Allows the user to edit their profile (change username and bio).
- /deactivate — Deactivates user account, removing their information from the database.
- /joke — Calls to the pokemon, superhero, and joke api to format and displays a joke to the user. The user can favorite the joke.
- /search — Allows the user to search for a pokemon or superhero and displays its stats if found.
- /pokemon/<ID> — Displays information about the pokemon with the given ID (taken from pokeapi). Displays error page if the ID does not exist.
- /hero/<ID> — Displays information about the hero with the given ID (taken from Superhero API). Displays error page if the ID does not exist.

List of templates we'll need:

- hero.html
- home.html
- landing_page.html
- pokemon.html
- search.html
- user_profile.html
- view_joke.html

Assignments of each task to each group member:

- Mahir:
  - Database files and setup
  - Helping Kevin with the python files if needed
- Karen:
  - HTML, CSS
  - A profile for the user when they are logged in & a template for the profile
- Ian:
  - User accounts & sessions
  - A landing page with a button allowing you to log-in to the website and sign-up for website & template for landing route
- Kevin:
  - Sqlite database along with flask app
  - Working on python method to deal out data