SnazzySnappers: Tanzeem Hasan, Ethan Sie, Linda Zhang, Nia Lam
SoftDev
P01: ArRESTed Development
2024-11-27
Time Spent: 3 hrs
TARGET SHIP DATE: {2024-12-16}

Program Components and Connections:

Frontend Components:

1. Jinja Templates - Updated as new data is requested by python
    a. /
        i. Users are able to enter the city where they want to see the weather of, in addition to other historical data of that location
        ii. Redirect to registration page if not logged in
    b. /registration
        i. Page where account creation and user log in takes place.
    c. /view_city
        i. Renders a heat index map of the area. Navbar on top that allows users to view precipitation levels, humidity etc.
        ii. Button to redirect to view climate history of the location
    d. /history
        i. Shows weather map of location through a timeline (slider)
        ii. Data table of yearly high and low temperature, precipitation etc.
    e. /natural_disaster
        i. Users are able to enter the city where they want to see recorded earthquakes, hurricanes, etc. and current disaster warnings
    f. /user_history
        i. Lists names of the user's previous ten searches alongside the time of search
2. Tailwind CSS - Frontend Framework

Backend Components:

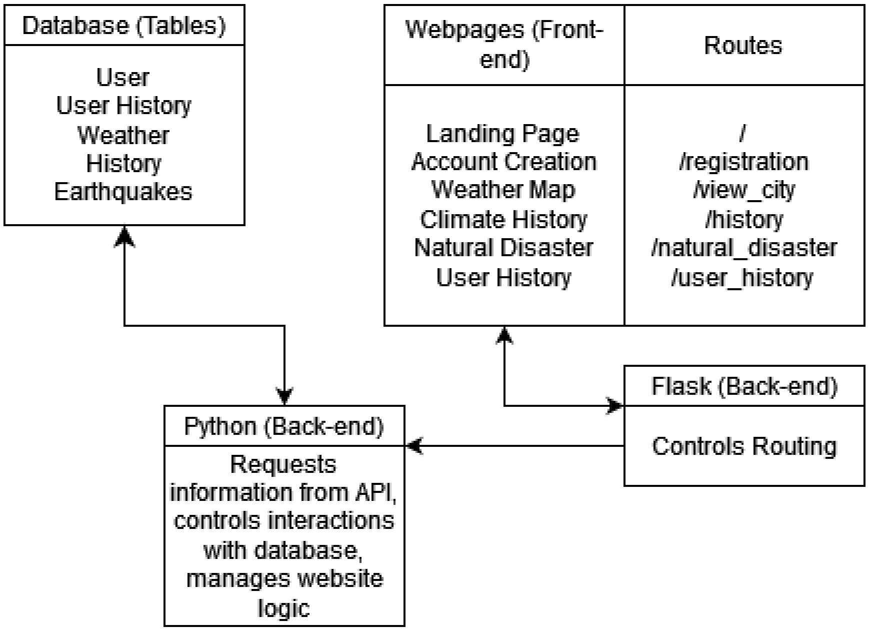
1. Flask/Python
    a. Allows the user to traverse different web pages when logged in. Python requests required information regarding weather, earthquakes, and the population of a specified location from APIs. From there, Python stores that data in a relevant database.
2. SQLite Databases - Stores information from APIs requested by Python
    a. user: will store user identification, password, name, and last login information

    b. user history: will store names of the user’s previous ten searches alongside the time of search
    c. weather: will store grid point information from open weather map API
    d. history: will store periodical climate data from visual crossing API
    e. earthquakes: store earthquakes, descriptions, magnitude etc.

Frontend Framework: Tailwind:

1. Tailwind CSS allows the writer to make use of existing utility classes as a shorthand when directly styling elements in HTML.
2. Tailwind has built in support for a responsive design, making it easier to create aesthetic buttons and sliders for this project.

Component Map:



*Inspired by Jobless_Monkeys component map from po0

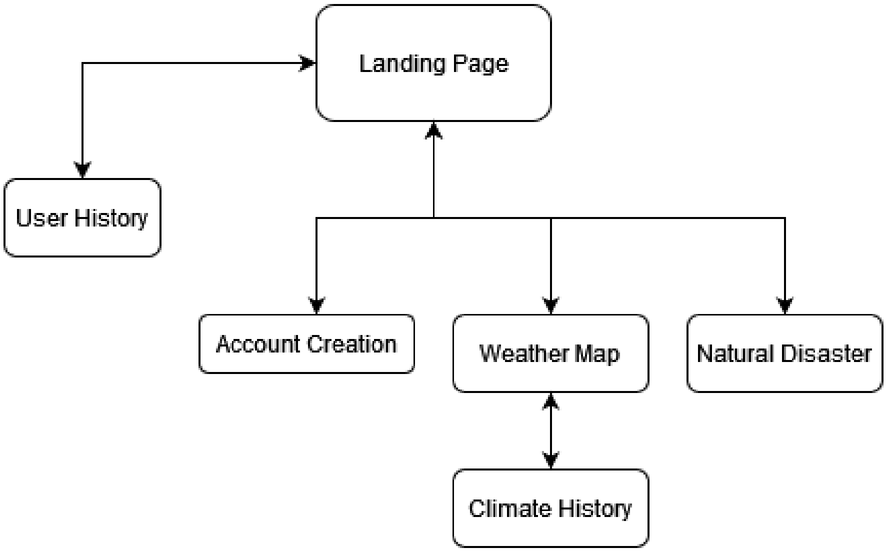
Database Organization:

1. User Table
    a. user_id (integer): unique identifier per user
    b. username (string): username chosen by user
    c. password (string): hashed password for security
    d. last_login (string {date-time}): tracks last user interaction
2. User History Table
    a. user_id (integer): unique identifies the user’s query
    b. search_type (string): distinguishes which database (weather, history, earthquake)
    c. location_name (string): name of city/area searched

    d. search_time (string {date-time}): timestamp of search

3. Weather Table
    a. weather_id (integer): unique identifier
    b. location_name (string): name of city/area
    c. latitude (float): latitude of location
    d. longitude (float): longitude of location
    e. temperature (float): temperature in celsius
    f. humidity (integer): humidity percentage
    g. precipitation (float): precipitation in mm
    h. wind_speed (float): wind speed in km/h
    i. timestamp (string {date-time}): time when weather was tracked
4. History Table
    a. history_id (integer): unique identifier
    b. location_name (string): name of city/area
    c. latitude (float): latitude of location
    d. longitude (float): longitude of location
    e. year (integer): year of the data
    f. avg_temperature (float): average temperature in celsius
    g. avg_precipitation (float): average precipitation in mm
    h. high_temperature (float): highest recorded temperature for the year
    i. low_temperature (float): lowest recorded temperature for the year
5. Earthquakes Table
    a. earthquake_id (integer): unique identifier
    b. location_name (string): name of city/area
    c. latitude (float): latitude of the location
    d. longitude (float): longitude of the location
    e. magnitude (float): magnitude of the earthquake
    f. depth (float): depth of the earthquake in km
    g. description (string): description/details of earthquake
    h. timestamp (string {date-time}): date/time of earthquake

Site Map:



APIs to use:

- Google Fonts:
    - API: https://developers.google.com/fonts/docs/developer_api
    - Github Card: https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_GoogleFonts.md
- OpenWeatherMap
    - API: https://openweathermap.org/
    - Github Card: https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_OpenWeatherMap.md
- VisualCrossing API:
    - https://www.visualcrossing.com/
    - Github Card:
- EarthquakeUSGS
    - API: https://earthquake.usgs.gov/fdsnws/event/1/
    - Github card: https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_EarthquakeUSGS.md
- WorldPop
    - API: https://www.worldpop.org/sdi/introapi/
    - Github card:

Tasks:

Tanzeem Hasan:

- Python routing between HTML templates
- Accessing information from WorldPop API and integrating with database
- CSS styling with Tailwind

Ethan Sie:

- Implement user history page and database
- Accessing information from EarthquakeUSGS API and integrating with database

Linda Zheng:

- HTML template design and CSS styling with Tailwind
- Implement functionality of Google Fonts API

Nia Lam:

- Implement user registration page and database
- Accessing information from VisualCrossing API and integrating with database