Green Graphs by **Team Bagel Pizza**

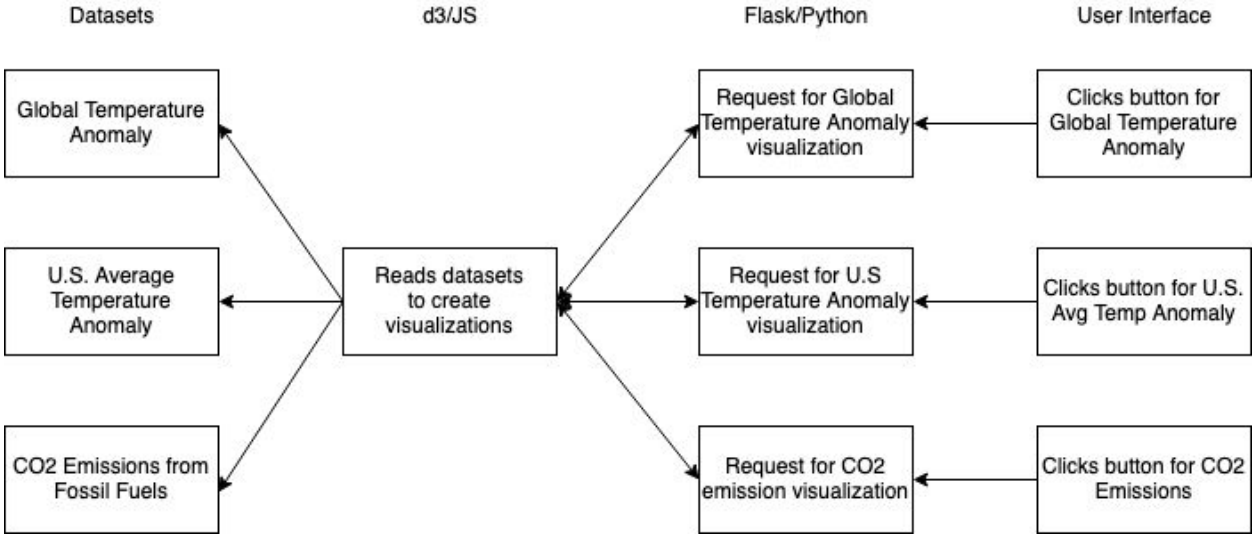
Alice Ni, Leia Park, Pratham Rawat, Hilary Zen

Our website's objective is to display climate change through graphs demonstrating the changes in temperature from 1901 to 2000, as well as CO2 emissions from fossil fuels globally. To use our site, the homepage will have three buttons labeled "Global Temperature Anomaly", "U.S. Average Temperature Anomaly", and "CO2 Emissions". When each button is clicked, a graph will appear that shows these data over time. The graphs are interactive, so sliders can be used to move through time. When the buttons are clicked again, the graphs will collapse. An additional button labeled "Compare" will show a graph comparing temperature anomalies and CO2 emissions over time.

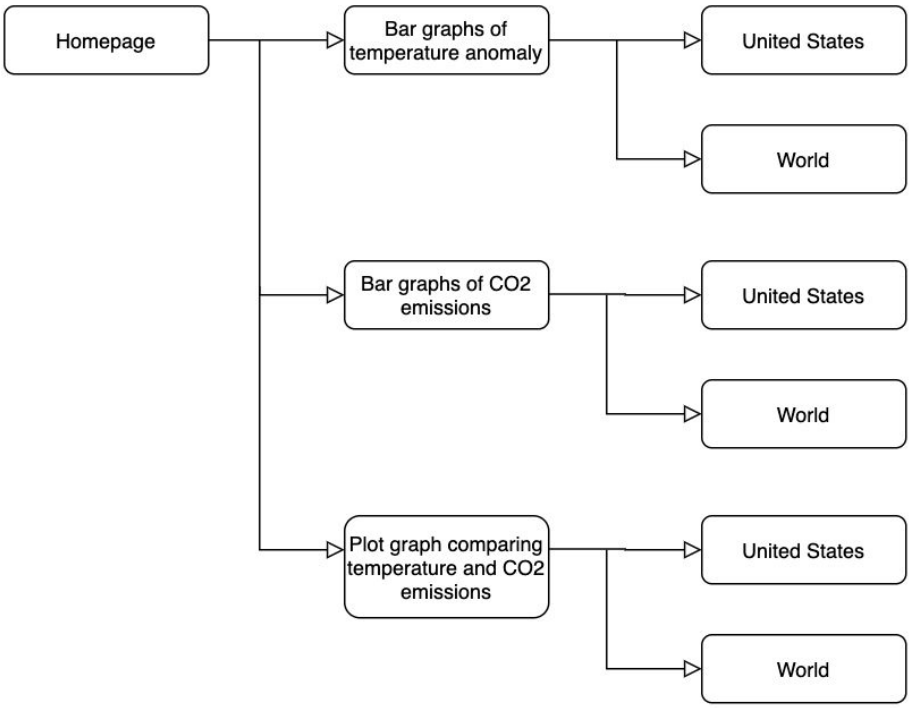
Roles:

Alice: Front-end, Bootstrap, JS/D3
Leia: Python, Database/CSV
Pratham: JS/D3
Hilary: Project Manager (prime minister), Flask

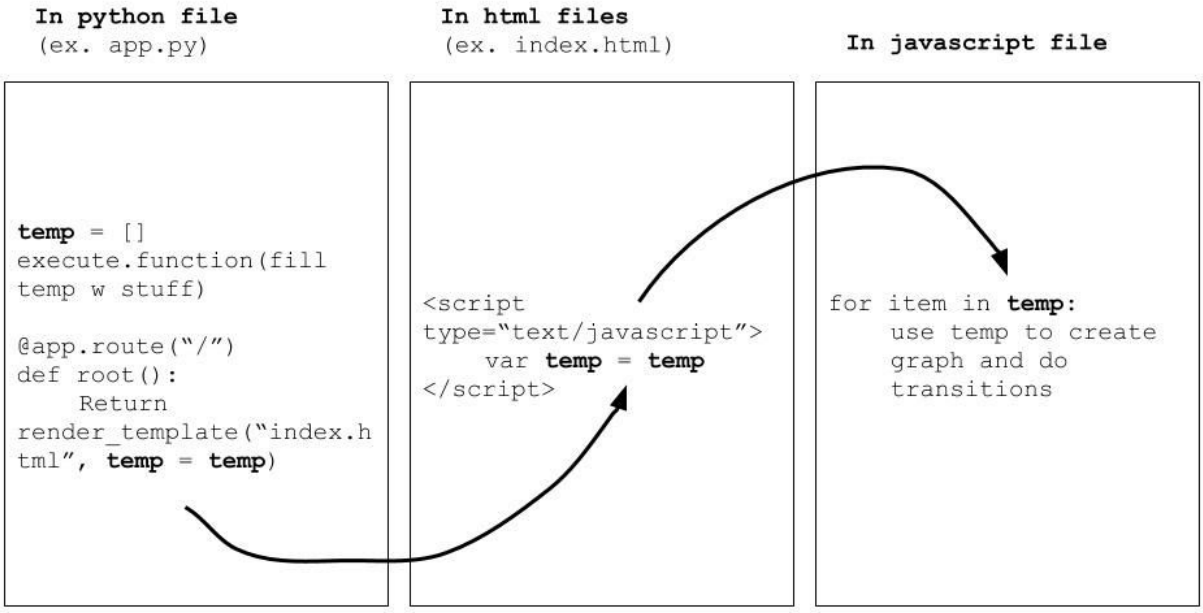
Component Map



Site Map



Breakdown of Data Transfer between Front & Back End



- The app.py file will read and organize the csv data and represent the data as variables.

- When executing the “render_template” function, the variables on the front-end will equate to the variables established in app.py
    - Ex. return render_template(“index.html”, temp = temperature…)
- For the d3 graph to display on the front-end, javascript is necessary
    - In the html file, a variable can be established for javascript by referencing to the variable obtained from app.py
    - Ex. <script type=”type/javascript”>
        var temp = {{temp}}
        </script>

Data Layout

Global Temperature Anomaly : JSON (will convert to CSV)
U.S Average Temperature Anomaly : JSON
CO2 Emissions from Fossil Fuels since 1751, By Nation : CSV

These datasets will be saved as files and built-in d3 CSV functions will read from these datasets to turn them into graphs/other data visualizations.

Front-end Framework

Bootstrap

- Because we are familiar with bootstrap, we decided it would be the best option for us.

Files:

- home.html
    - Will serve as the sole page for our site (unless otherwise decided)
    - Buttons that allow the user to generate graphs that display the requested information
    - Clicking button again will collapse the graph
      (it will either be this, or having multiple pages with each page dedicated to a different dataset. The benefits of multiple pages are that we can display more information, and not just a graph, but idk if we’ll have time for that)

Backend Framework

- app.py
    - Contain flask to pull html files
- data.py

    - Will read and organize csv files into easily manageable data as lists or dictionaries if needed
- data.js
    - Takes data and uses D3 to create various graphs