Proposal — Team095-OAR

Project Title: eco-journey

Project Summary:
Our project puts together historical weather and geographic data from the United States into a single platform built to give users better insights into tourist destinations in the US. By combining data, users can make more informed travel decisions by looking at important info like the best times to visit, typical weather conditions, etc..

Description of Application:
This website will access our database which has various weather and geographical information such as temperatures, precipitation, humidity, and wind which can be utilized to determine a “grade/score”(i.e. A-F) that the user can then use to make decisions about their travel plans.

Creative Component: Dynamic Grading Model
Allows the user to compare 2-4 possible tourist destinations side-by-side. Users will select from a list of features that are important to them (lack of rain, heat, etc.) and rank features by importance. Afterwards, the model will calculate a score by assigning weights for each feature and computing some numerical final score, which will be reported back to the user on an A-F scale (with some kind of tie-breaker so that no two locations receive the same score).

The model will use historical data in order to predict the expected conditions at the expected time of year the vacation will take place. The model will also output a visualization (wheel graph, bar graph) visualizing the differences between the ranked features. Each feature will be colored on a red-blue gradient (with red being the weakest feature and blue being the strongest.)

[Optional] The model will take into account climate change when calculating the scores (rising sea levels, rising temperatures, increased natural disasters, etc.)

[Optional] Input the results into ChatGPT via OpenAI API to provide a brief description of each location, evaluating its strengths and weaknesses.

Usefulness:
Existing weather apps like The Weather Channel only provide current weather data for a location, which can be very inconsistent. For example, a standard tourist looking to visit Chicago would just take a look at the current weather and predict the forecast for the duration of their stay to determine whether they should visit. However any local knows that the weather forecast is rarely accurate, which is where our implementation with historical data would be more accurate.

Our solution would also offer the ability to compare this historical data for multiple locations. This website would be useful for tourists who want to see which locations would be best to travel to for their vacation, as well as researchers who want to see past weather/climate data for specific cities. Finally our algorithm would also offer a relative grade for each location, so users don't need to analyze the data themselves and just choose the best ranking

Dataset and Realness:
NOAA Global Historical Climatology Network Daily Dataset (link)
EPA Air Quality System (AQS) Data (link)
Simplemaps U.S. Cities Database (link)

For our project, we will use three real-world datasets. The first will be the NOAA's Global GHCNd dataset. The plan is to get it via a bulk download from their servers, but the data is in a fixed-width unique format (.dly), not CSV. As each line represents a month's data for one weather element we'll have to programmatically parse it back into CSV or some DB readable format. Our second source, the EPA's AirData, is available only as individual CSV files broken down by year so we'll need to merge them. This means we'll need to make a decision about how far back to go and how much to merge. Lastly, the Simplemaps U.S. Cities Database, which has geographic data, is a direct CSV download from their website. This one is the most straightforward and doesn't need much initial effort to work with.

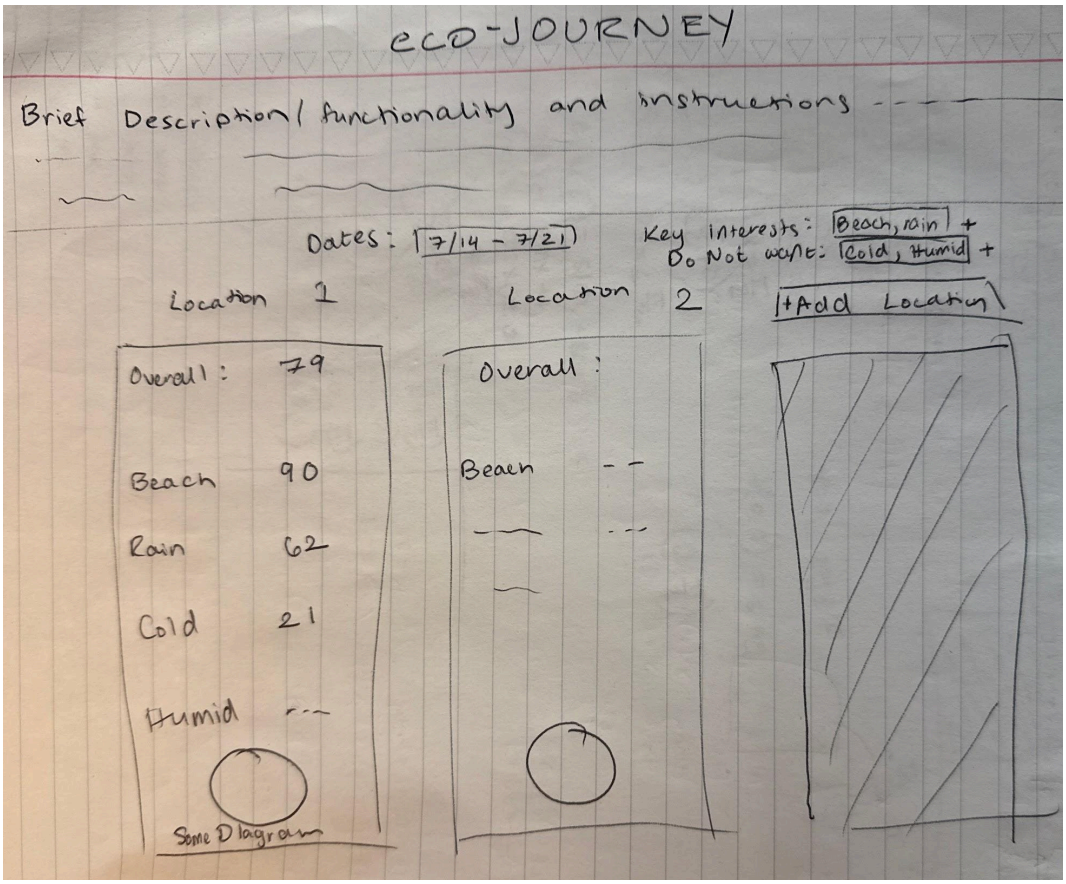
These datasets are also all going to need scaling and processing. For example, for the NOAA GHCNd dataset, after parsing the files, we'll need to convert the raw integer values to units like Fahrenheit and inches. For the EPA AirData, our primary processing task will be to select relevant columns and ensure data types are correct. The Simplemaps cities CSV, with over 30,000 rows, is already structured well and will serve as our primary location table.

Since these datasets lack a common ID column, our linking strategy is going to be on geospatial data using the latitude and longitude coordinates. Our application's cities table (from Simplemaps) will be the central reference. The goal is when a user selects a destination like "Chicago," the application will use its stored latitude and longitude to perform a location-based query. It will search the NOAA station metadata to find the nearest weather station IDs and query our merged EPA data for the closest air quality monitors. We can then associate the correct historical weather and air quality records with each tourist destination, effectively joining the data based on location and proximity instead of a shared key.

User Inputs:
At its core, the site allows any user to immediately engage with our primary search and read functionalities. A visitor can look up any U.S. city to view its detailed historical weather and environmental profile or use the main comparison tool to analyze locations side-by-side. This comparison feature is highly interactive; users submit a form with their planned travel dates and rank the importance of personal preferences like temperature, precipitation, and air quality to generate a custom-tailored analysis.

To unlock a better experience, a user can create a personal account through a simple registration form. Once logged in, they gain access to a full suite of CRUD capabilities centered around saving their research. The key feature for registered users is the ability to create a "Trip Plan" by saving the results of a generated comparison, which stores their selected cities, dates, and preference weights in our database. From a personal dashboard, users can later read and revisit all their saved plans, and they have the flexibility to update a plan by modifying its parameters or delete it entirely if it's no longer needed. Furthermore, users have full control over their account, with the ability to update profile information like their password or permanently delete their account and all associated data.

Low Fidelity Mockup:



Project Distribution:

Our team's workflow is split between backend (Aditya, Ali) and frontend (Michael, Ojas). The backend team will handle data processing, database management, and the scoring API, while the frontend team will build the user interface and visualizations. All members will collaborate on database-related tasks (SQL, schema design, etc.).

<u>Backend Development</u>

Aditya: Data Pipeline & API

Aditya will lead data engineering by writing Python scripts to parse, merge, and clean the NOAA and EPA datasets. He will then design the database schema and build the core RESTful API using Flask, creating key endpoints like /api/cities and /api/compare.

Ali: Scoring Algorithm
Ali will develop the dynamic grading model. His work involves retrieving historical data and implementing the weighting algorithm that processes user preferences to calculate a final A-F score. He will integrate this logic into the API and lead the potential OpenAI implementation.

Frontend Development
Michael: UI & Visualization
Michael will translate our mockup into a functional UI, managing the layout, styling, and core components. He will also implement the data visualizations using libraries graphically representing location comparison data.

Ojas: State Management & Logic
Ojas will manage the frontend's state and client-side logic. He will handle all user inputs, communicate with the backend API by making fetch requests, and ensure the UI is dynamic and responsive to user interaction and incoming data.