Tourism Recommendation System

Mithali Shashidhar
Ananya Nair
Tanushree Yandra
Anish Dixit

Background

- Tourism Recommender sites are commonly used to get an idea of places worth visiting in a particular location. (TripAdvisor, Google Maps etc.)
- Current Tourism Recommender apps/websites provide information about good places to visit, but it is not in real-time.
- Factors like current weather conditions can play a big part in determining what place to visit.
- For e.g., on a rainy day, you might not want to visit a beach.
- Our system tries to blend popular tourism spots with weather conditions to curate a list of good places to visit.
- We also give additional information such as the rating, price level and opening hours for some places, which can provide a holistic experience for visitors.

Data Sources

Google Maps API

- Using Maps API, we pull request details corresponding to the user entered query.
- This data is acquired in a JSON format and stored as a dataframe.
- Data features include location, type of establishment, date, time, coordinates, etc.
- Limitations:
    - Price level variable not available for all types of places.
    - API has a limit on the number of calls that can be placed daily.

Meteostat Weather API

- The Meteostat Python library provides a simple API for accessing open weather and climate data.
- Meteostat API helps get weather details for a particular location and time period.
- Data features include temperature, humidity, precipitation, weather category etc.
- Limitations:
    - Weather data not available beyond eight days from the current date.
    - Some weather variables return a null value beyond 3-4 days from the current date.

User Interaction

The User inputs a variety of parameters to get the response tailored to them:

- **Location:** The user is asked to enter what is the place for which they want recommendations. If left blank, the user's current location is picked up.
- **Date:** Date of intended visit
- **Time:** Time of intended visit
- **Type of Establishment:** What kind of place does the user want to go? Cafes, Beaches, Parks, Movie Theaters, Museums etc.
- **Budget:** If the user has a max. Price level in mind for the places.
- **Rating:** The user can choose to enter a minimum rating that the place must have.

Based on these entries, and our recommendation engine which adds in weather related logic, a curated list of places to visit is returned to the user.

Use Case 1 - Ellie

Assume one of our users, Ellie, is going to Seattle on a business trip tomorrow. She will be staying in Downtown, and would like to visit somewhere (preferably high-end cafes and restaurants) in the evening around 6:30 PM. Based on her requirements, she would fill the data fields as follows:

- **Personal Preference** - Cafes and Restaurants
- **Desired Date of Travel** - Tomorrow's date
- **Desired Time of Travel** - 18:30
- **Minimum Preferred Rating** - 4.5
- **Maximum Preferred Price Level** - 5
- **Location** - Downtown, Seattle

Ellie would not only receive recommendations based on her preference, but also based on the weather. Ellie could then choose to explore other places as well.

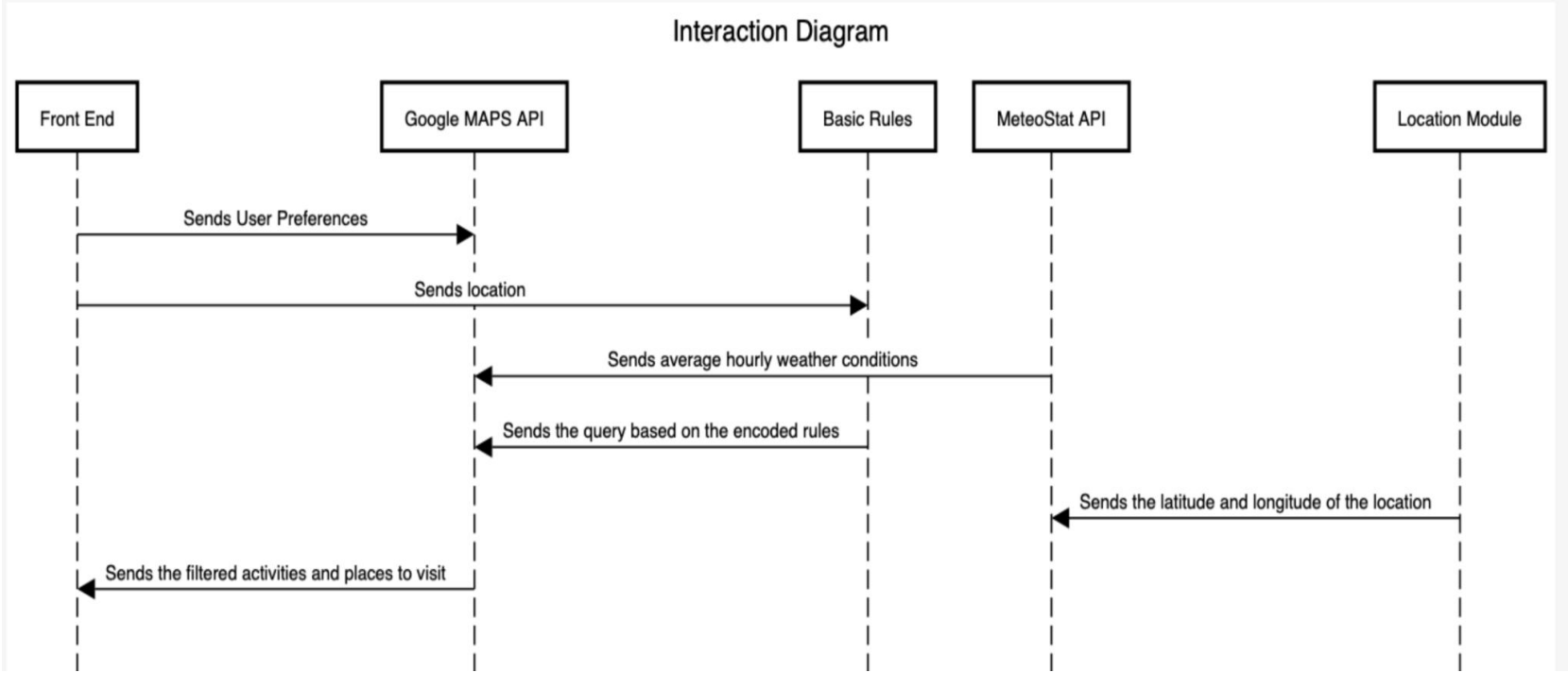
Use Case 2 - Andrew

Now let’s assume that one of our users, Andrew, has been living in Seattle for the last one year. With Seattle's weather being so erratic, Andrew’s plans don’t always go as planned. Andrew wishes to go explore places around him that are appropriate for that day’s weather. Let’s say Andrew wants to explore places around him at noon, two days from now, with a minimum rating of 3.5 and a minimum budget. Based on his requirements, he would fill the data fields as:

- **Personal Preference** - None
- **Desired Date of Travel** - Day after tomorrow’s date
- **Desired Time of Travel** - 12:00
- **Minimum Preferred Rating** - 3.5
- **Maximum Preferred Price Level** - 1
- **Location** - (leave blank to use the current location)

Andrew will then get a list of places he can explore by category based on his desired date’s weather conditions.

Design


Interaction Diagram
Front End
Google MAPS API
Basic Rules
MeteoStat API
Location Module
Sends User Preferences
Sends location
Sends average hourly weather conditions
Sends the query based on the encoded rules
Sends the latitude and longitude of the location
Sends the filtered activities and places to visit

DEMONSTRATION

Lessons Learnt and Future Work

- Functional Specifications can go a long way in ensuring clarity of communication between clients and developers and ensure everyone is on the same page.
- Test Driven Development is an important technique to ensure your project is on the right track with respect to completeness and reproducibility.
- Continuous Integration can greatly help test progress regularly rather than just towards the end.
- While using Github (or any version control), maintaining separate branches for each developer is a good way to avoid mix ups of code.
- Exception Handling is underrated but important.

Future Work:

- Improve the recommendation logic and making it more complex by adding in multiple parameters.
- The UI can be made fancier by using templates, adding pictures, a google maps link etc.
- We can try extending our date range beyond the current eight days to allow users to make plans much ahead of time.

Questions?