4.3 Proposed Design

4.3.1 Overview

Our design will consist of three major subsystems:

- Front-end user interface that will enable the users to:
    - Specify the categories of points of interests (POI) to be visited.
    - Specify the distance constraints for the trip.
    - Specify the starting-category.
    - Display the recommended trajectory for the visit as well as the recommended hotel (i.e., starting point).
- Back-end server
    - Store data
    - Generate route containing POIs
- "Middleware" that will connect the front-end and back-end.

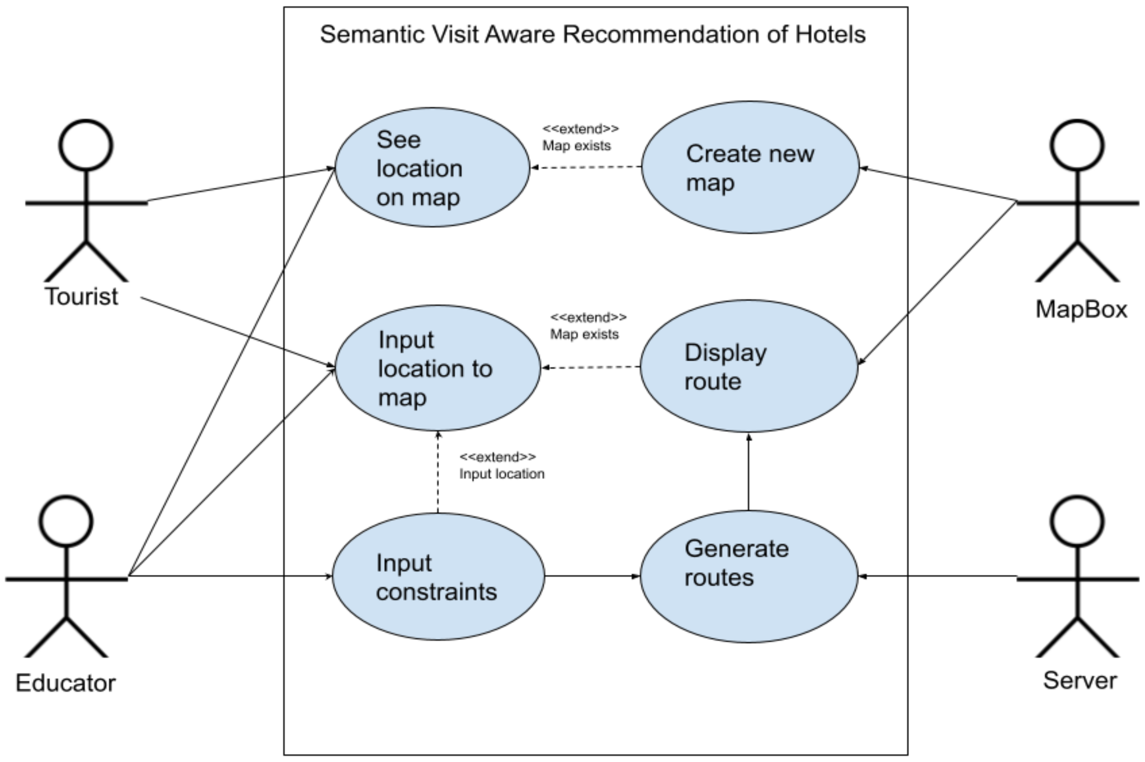


Figure 1. Use case diagram

4.3.2 Detailed Design and Visual(s)

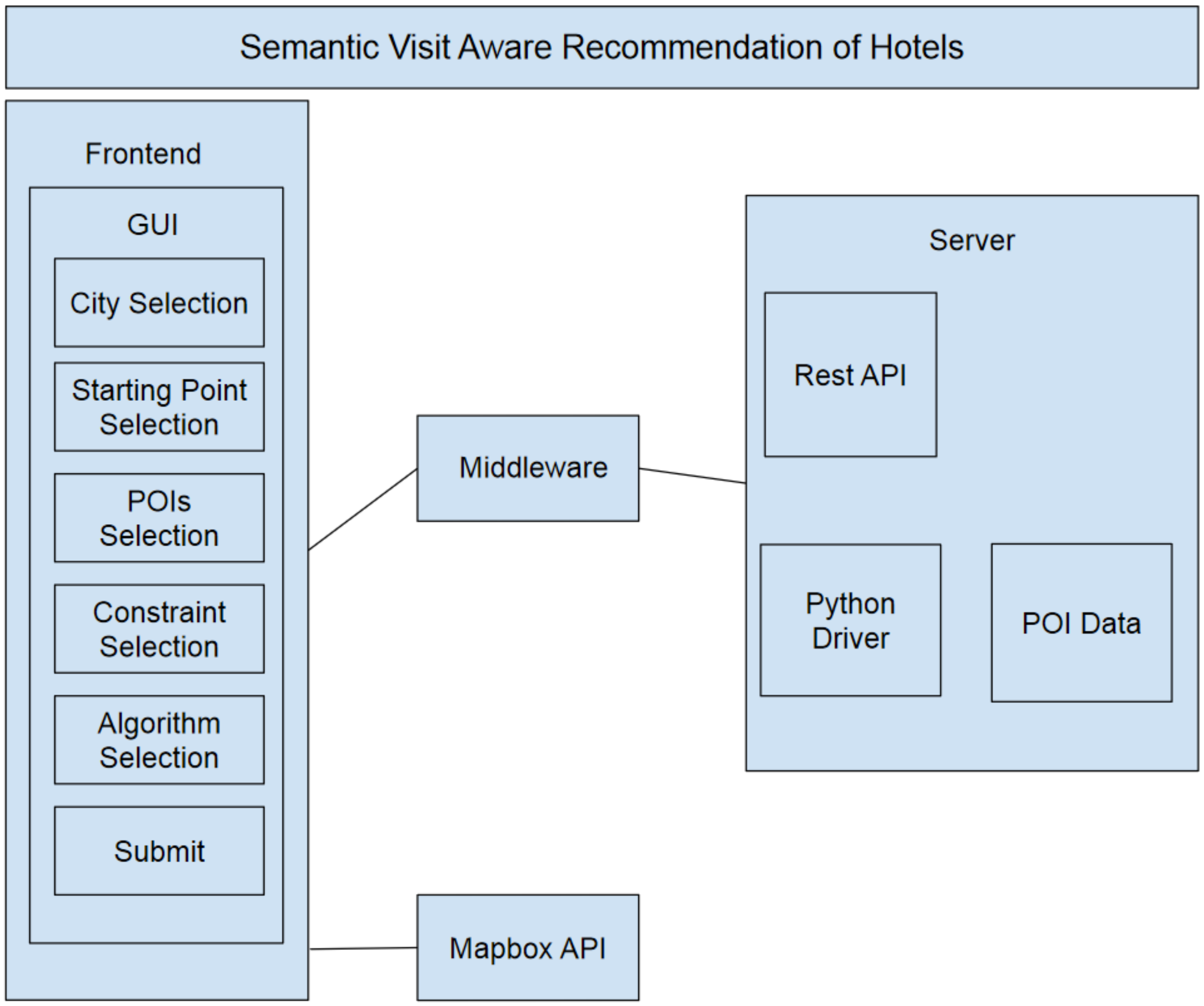


Figure 2. Block diagram

To best describe our design, we will describe our sub systems separately. Once the design of the subsystems is complete, we can piece together the whole system.

Front-End

The user interface for our application will primarily be used in a web browser on a computer. The web application will be just a single page and made in React and TypeScript. The user experience needs to be clean and easy-to-use. This will be achieved by having a simple layout and by keeping related information together on the screen.

Back-End

The server will be used to store the given POI data which will be used to execute the algorithms and generate the path with POI. The backend will also communicate with the front end to receive constraints such as starting point, preferred POI, etc... from the user and provide the constraints to the algorithms.

Middleware

The middleware will be used to connect the frontend and backend together so we can efficiently send/receive requests made by the user in the UI and process them with the Python algorithms and location data stored in the backend.

Routing Algorithms

For our project, we received four routing algorithms. The algorithms are designed to find an efficient route from a starting point to different destinations specified by the categories given by the user. The algorithms designed to provide a best path are NSS-kDPQ and ESS-kDPQ. We were also provided a Dijkstra algorithm and Random Walk with Restart.

4.3.3 Functionality

When a user opens the web application they will be shown a map of the city that they would like to explore and plan their stay-location (e.g., a hotel) as well as a sequence of PoIs to be visited. The default/initial implementation will focus on New York City. In addition, they will be presented with a form that will allow them to specify constraints and other information for route generation. This can be seen in Figure XX. below:

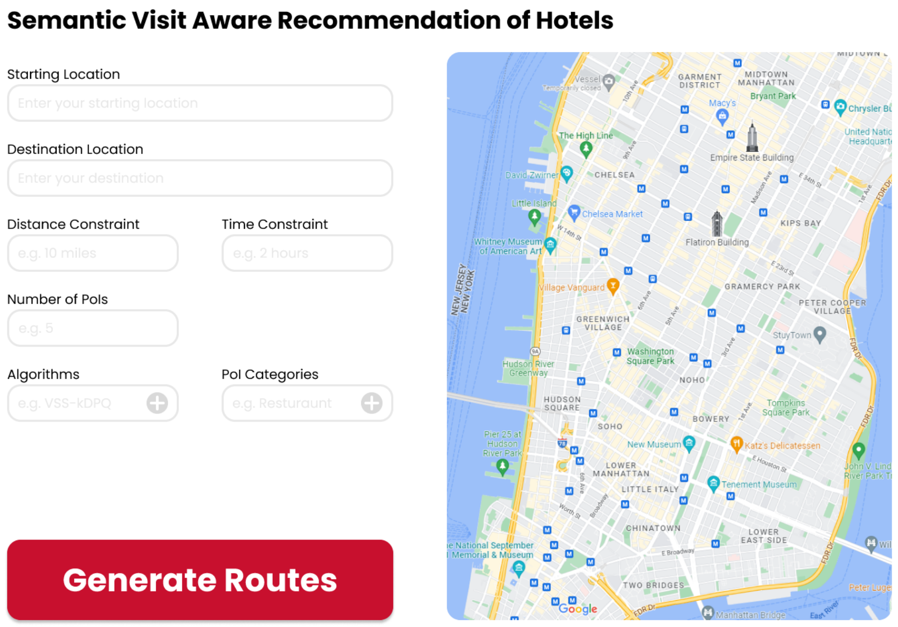


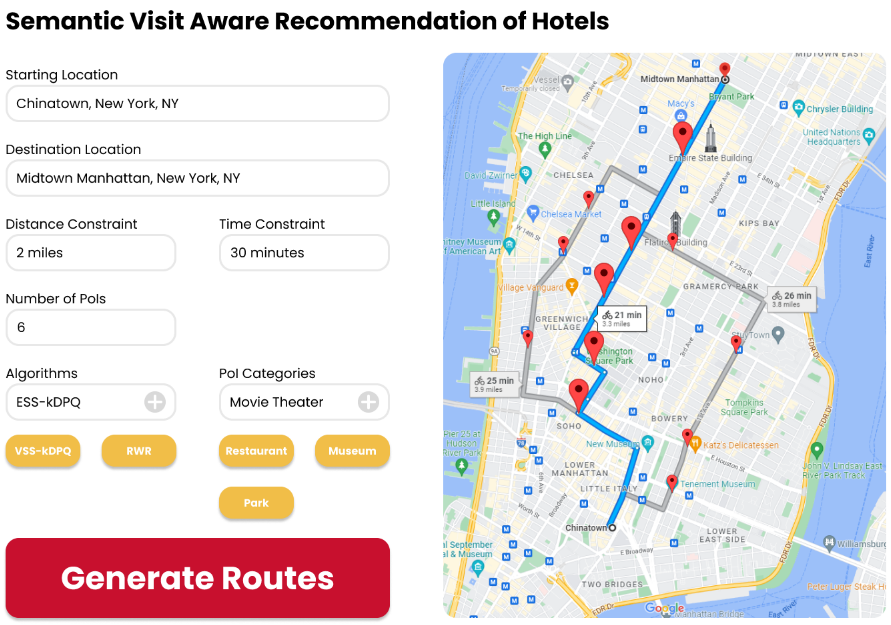
Figure 3. Initial User View of Front-End

Users will be able to enter only one value for each of the inputs in the form, except if the form element has a plus sign inside the input field. This plus sign indicates that the user can enter multiple values into the input field, and once entered, the values will appear below the input field so the user can see what has already been entered.

Once the form is filled out, the response by the system will be executed in a manner that will generate routes satisfying the given constraints. To do so, the user needs to click on the “Generate Routes" button. This will send the user-entered information to the backend where, as shown in Figure 2, the Python driver will be triggered. This, in turn will generated the following sequence of activities:

1. Access the file containing the POI data (CF. Fig. 2)
2. Invoke the route generation algorithm (selected by the user’s request)
3. Generate the routes that will be stored in a JSON file.
4. Send the JSON to the MapBox API to visualize the routes.
5. Upon completion, an event will be generated that will reflect the view on the map of the UI.

After this sequence of events, the web application will be updated to look like Figure 3.



Semantic Visit Aware Recommendation of Hotels

Starting Location: Chinatown, New York, NY

Destination Location: Midtown Manhattan, New York, NY

Distance Constraint: 2 miles

Time Constraint: 30 minutes

Number of PoIs: 6

Algorithms: ESS-kDPQ

PoI Categories: Movie Theater

Figure 4. Front-End View After Route Generation

4.3.4 Areas of Concern and Development

Our primary concerns for delivering a product that meets our user and client needs are the following:

- We will need to learn how to use the MapBox API to incorporate visuals and also provide routes from our routing algorithm
- We will have to learn how to use the React framework to create our frontend
- We are not certain on how to develop effective integration testing to test the system’s behavior.

To address the above concerns we are going to do the following:

- We will read the MapBox documentation and test it with our solution
- We will read the React documentation and help others on the team when needed
- We will ask our client how to go about developing effective integration tests.

4.4 Technology Considerations

React - React is modern, has lots of developer support,and allows you to create components that can be used to cut down on coding time. However, it can be very slow if not well optimized. Because it is being updated so often, it can have poor documentation. There are some viable alternatives, Angular for example, but these are less popular.

MapBox API - The MapBox API is documented extensively online, making it easy to learn. The load times are quick and access to data is easy. However, the system can be slightly complex and has a steep learning curve. Alternatives for other APIs are often too simplistic and don't offer as much functionality (such as OpenStreetMaps), or have stricter API limits (such as Google Maps), or have MUCH steeper learning curves (such as ArcGIS Online). We must have a mapping API in order to display paths so there are no alternatives to having a mapping API.

[Back-end language/framework] - Java is a flexible, relatively high speed language that everyone in the group has lots of experience with. Using it to work with the CSV file and generate routes would be straightforward and efficient. The downside is that we have to have communication between these different parts of our project, which makes our design more complex. The alternative would be to use React to do these things directly, but that would be more inefficient and make our project more monolithic in design.

4.5 Design Analysis

We have not implemented or tested this design at this time, so we do not know if our design will work.The testing and implementation of our design will occur in the next semester.