Semantic Visit Aware Recommendation of Hotels

•••

sdmay23-34

Members: Dylan Hampton, Zachary Garwood, Thomas Frohwein, Joe Zuber, Britney Yu, Kevin Knack, Nathan Schenck

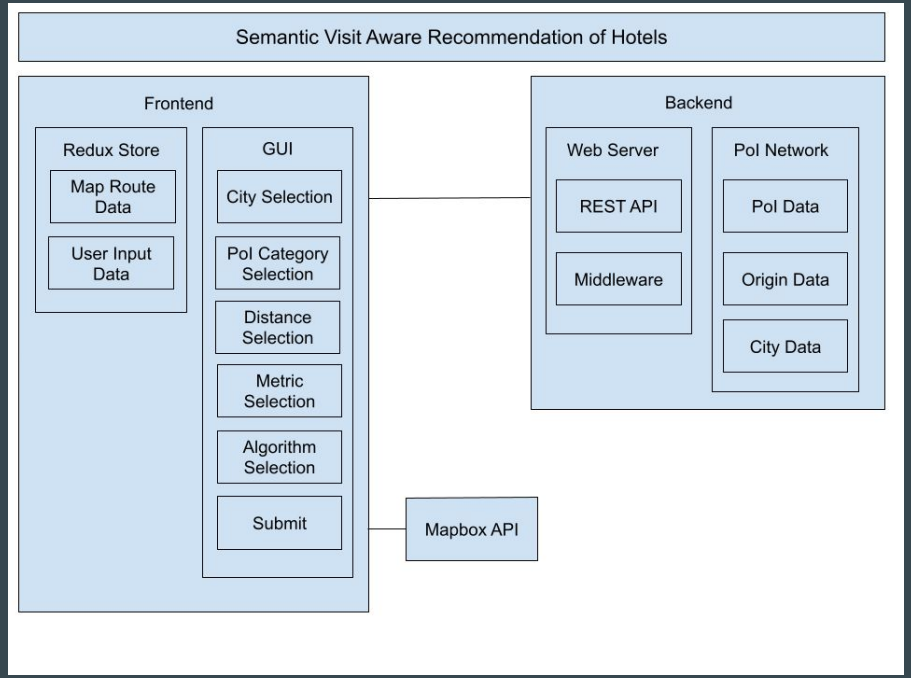
Client/Advisor: Goce Trajcevski

Introduction

- Routing applications (eg. Google Maps) give users the ability to easily find routes to selected destinations
- Booking applications (eg. Booking.com) give users the ability to find hotels in a desired location
- Neither of these tools can show a user what hotels can be used as starting points for routes to destinations which fit desired categories
- We have developed a visual prototype system that does just that:
  - Allows a user to select desired categories (museum, statue, park, etc.)
  - Generates a routes from an origin location (hotel or Airbnb)

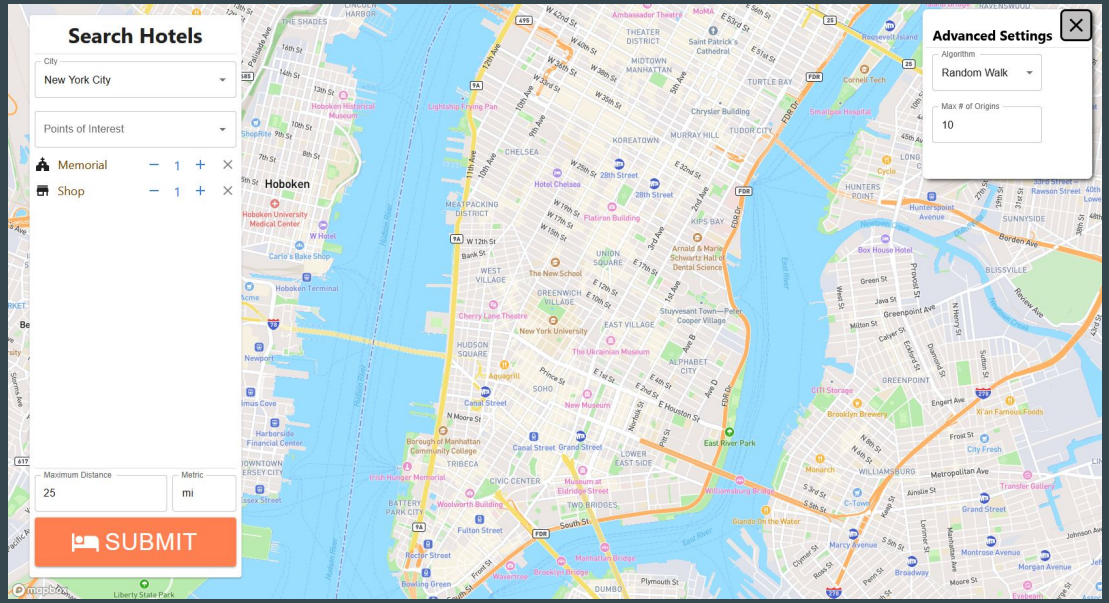
Implementation Architecture

- Frontend
  - Takes user criteria as input
  - Communicates with backend to run algorithm(s) based on criteria
  - Interacts with the Mapbox API, shows results directly on map
- Backend
  - Holds the PoI data
  - Runs the desired algorithms
  - Returns path data to Frontend
  - Returns best starting point for most diverse path



Implementation Architecture (Frontend)

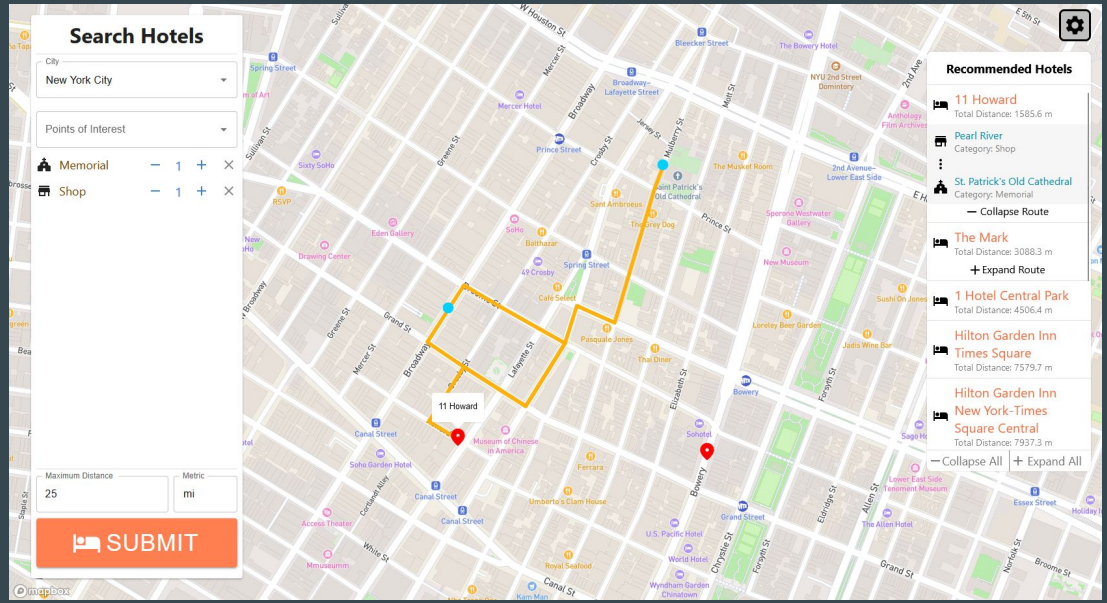
- React with TypeScript
- Redux Store to hold state of application
- User-defined criteria
  - Mandatory
    - city selection
    - distance constraint
    - metric constraint
    - category constraint(s)
  - Optional
    - algorithm choice
    - max number of hotels



UI Design

Implementation Architecture (Frontend)

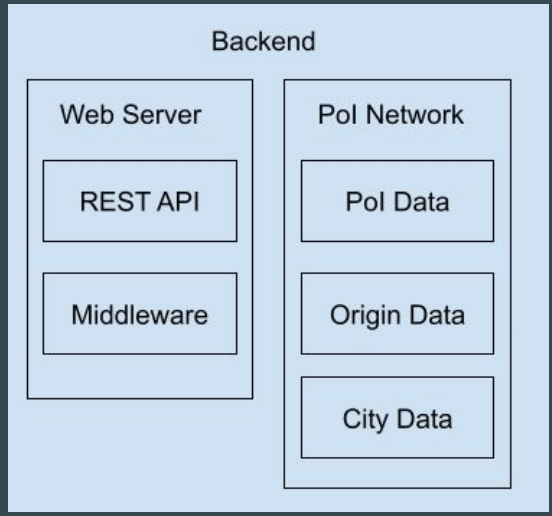
- Recommended Hotels Dropdown
  - Click on Hotel to have map display route and PoIs
- Red pins - hotels
- Blue dots - PoIs



UI Design

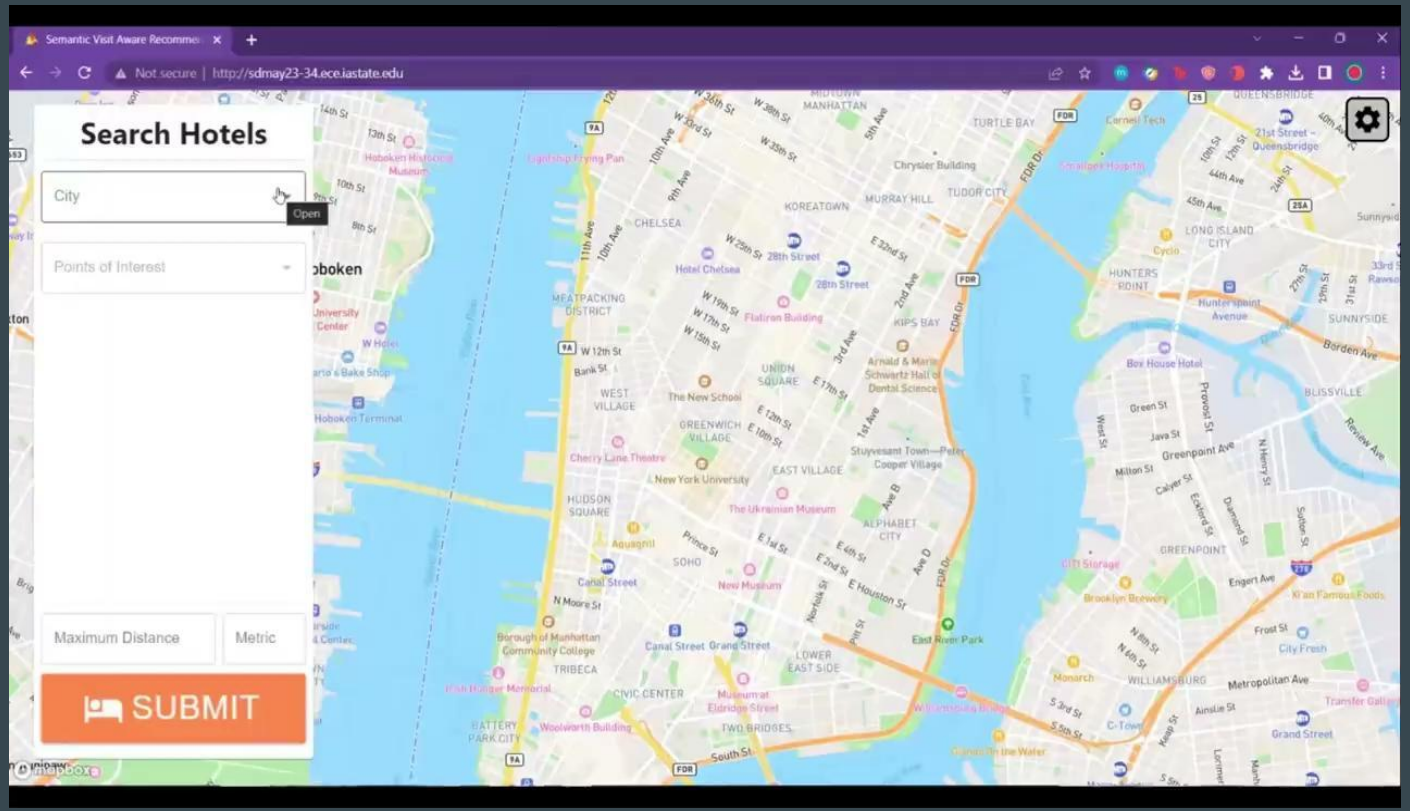
Implementation Architecture (Backend)

- Written in Python (Flask)
- Apache HTTP Server
- Store client's algorithms to generate routes
- Store PoI Network used by route generation algorithms
- API for communicating with frontend
- When user wants to generate route
  - Invoke the route generation algorithm (selected by the user's request)
  - Generate the routes that will be stored as a JSON object
  - Send the JSON to the MapBox API to visualize the routes.



Backend block diagram

Video demonstration

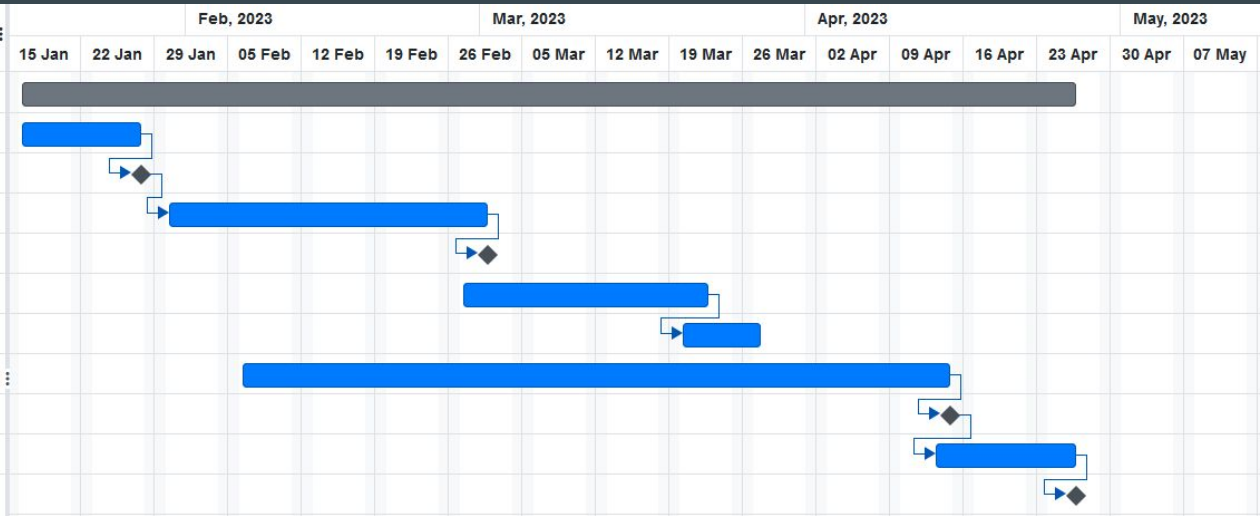


Project Plan - Frontend

KEY

- Major Section
- Task
- Milestone

Name	Start Date	End Date
Front-End UI	Jan 16, 2023	Apr 26, 2023
Figma Design	Jan 16, 2023	Jan 27, 2023
Set Up Code Base (Git, React Project)	Jan 27, 2023	Jan 27, 2023
Set Up Base UI	Jan 30, 2023	Mar 01, 2023
Basic UI Setup	Mar 01, 2023	Mar 01, 2023
Implement Redux for State Management	Feb 27, 2023	Mar 22, 2023
Connect to Backend	Mar 20, 2023	Mar 27, 2023
Implement MapBox	Feb 06, 2023	Apr 14, 2023
Implement Visualization in MapBox	Apr 14, 2023	Apr 14, 2023
Get User Input to Change Map	Apr 13, 2023	Apr 26, 2023
Visualize Routes in MapBox	Apr 26, 2023	Apr 26, 2023



Frontend Gantt Chart

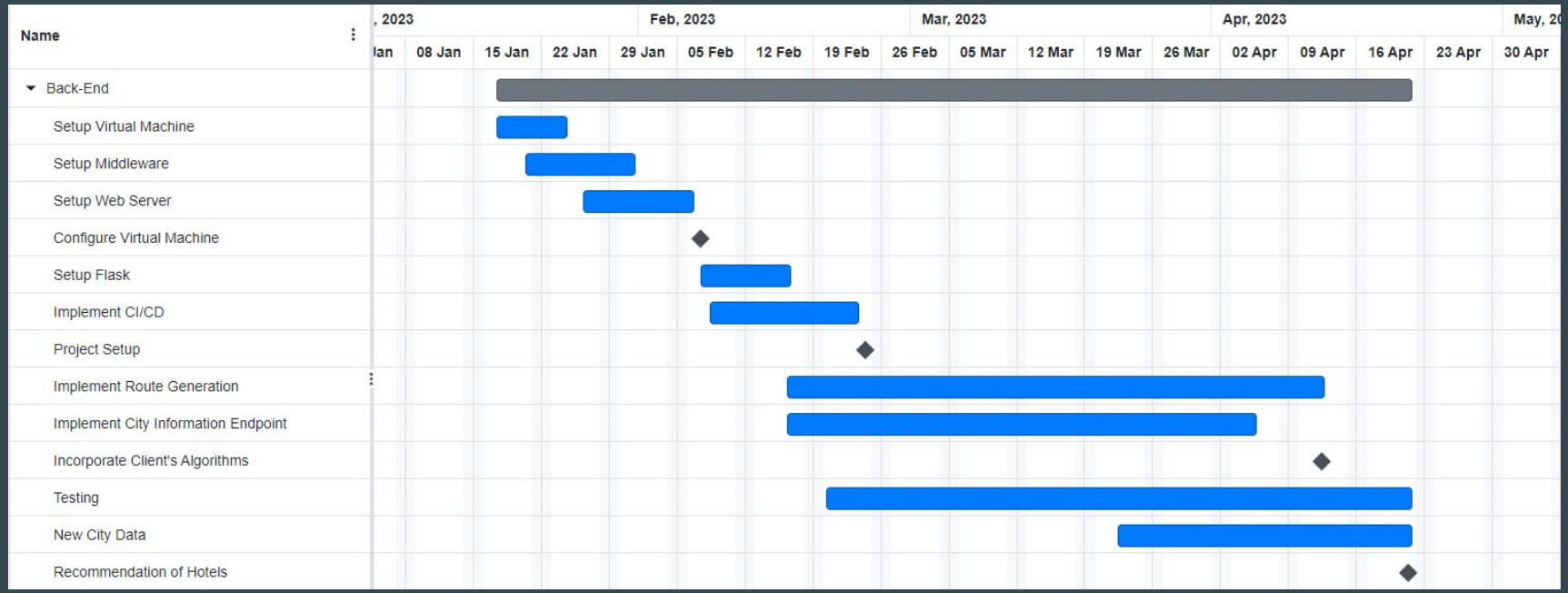
Work Accomplishments - Frontend

Objective	Tasks	Milestone
Basic UI Setup	● Figma Design ● Setup Codebase	UI is Setup
Implement Visualization in MapBox	● Implement Redux for State Management ● Connect to Backend ● Implement MapBox	Visualization is Successfully Implemented in MapBox
Route Visualization in MapBox	● User Input to Change Map	Routes are Successfully Visualized on Website

Project Plan- Backend

KEY

- Major Section
- Task
- Milestone



Backend Gantt Chart

Work Accomplishments - Backend

Objective	Tasks	Milestone
Configuring the VM	• Setup Virtual Machine • Setup Middleware • Setup the Web Server	• VM is Successfully Configured
Project Setup	• Setup Git • Setup Flask • Implement CI/CD	• Backend of Project is Setup
Incorporating the Client's Algorithm	• Implement Route Generation • Implement City Information Endpoint • Fixing any Issues that Arise	• Client's Algorithm is successfully Implemented
Recommendation of Hotels	• Testing • Recommendation of Hotels • New City Data	• Successfully Recommends Hotels • Successfully Recommends POIs • Added New City

Key Contributions

Joseph Zuber (Backend)

- Added setup code which sped up requests to backend
- Provided backend extension to implement Chicago (and possible future cities)

Nathan Schenck (Frontend)

- Added origin marker generation on route data response
- Created hotels/routes features in sidebar

Dylan Hampton (Frontend)

- Added PoI and route drawing functionality
- Added zoom functionality when routes are returned / are selected

Kevin (Backend)

- Fixed New York dataset
- Testing for backend functions

Key Contributions (Continued)

Zachary Garwood (Backend)

- Implemented route generation and city information endpoints
- Created Chicago dataset

Thomas Frohwein (Frontend)

- Implemented Redux to hold state of application
- Setup asynchronous calls to communicate with backend

Britney Yu (Backend)

- Restructured the project into separate packages and modules
- Identified issues with the categories PoIs through testing

Challenges and Solutions

Frontend

- Challenge: Issues with storing state of user input for submission
  - Solution: Implementing Redux to help store state information in slices

Backend

- Challenge: Java Python drivers did not offer the functionality we needed
  - Solution: Migrating our project to Flask
- Challenge: Very slow response times
  - Solution: Calculating graph setup information once for each city after the first time the city is requested, and making it persistent between requests
- Challenge: Missing PoI dataset
  - Solution: Talked with client to retrieve necessary dataset

Future Work

- More cities could be added
- Existing cities could be perfected
  - With more time and resources, PoI categories could be more accurate and more numerous
- Travel time between PoIs in a route could be calculated and shown to the user
  - To fully support this, multiple modes of transportation could be added
- Hotels/Airbnbs and PoIs could be further fleshed out:
  - Ability to view photos or information about the selected PoI
  - Links to view information or book a selected hotel/Airbnb could be provided
- Additional constraints could be added for route generation:
  - Cost of hotels
  - Availability of rooms in hotels
  - Requiring the inclusion of specific PoIs (eg. the Brooklyn Bridge) rather than categories

Conclusion

- Successful visualization of our client's algorithms
- Fulfilled all functional and nonfunctional requirements
- Application can easily be built upon in the future