Cabana - A Real Estate Search Engine

Madhushree Kumari
Rucha Sathe
Sakshi Nihatkar
Yuchen Wang
Ziya Ye

Problem Statement

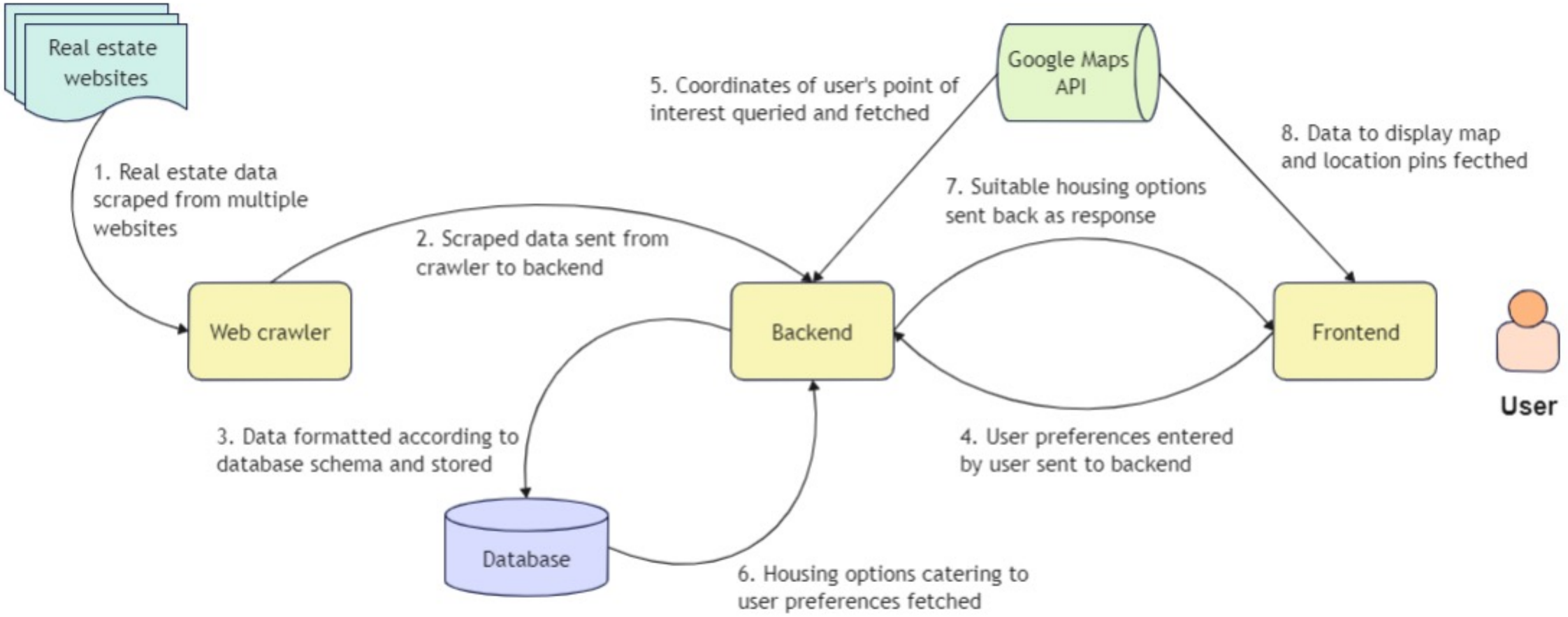
- Spend a long time searching for housings
- Searched on Zillow and then Airbnb and then ... (other housing platforms) for **affordability** but **no proximity**
- These sites lack a crucial element: **commute time!**

Our Solution

A real estate search engine that allows you to select housing on the basis of commute time to your workplace... school... or literally anywhere!

What's the highlight, you ask?
You DON'T have to switch to Google Maps to find the commute time. We give it to you! You can customize your commuting choice from driving, biking, and walking.

Proposed System Architecture

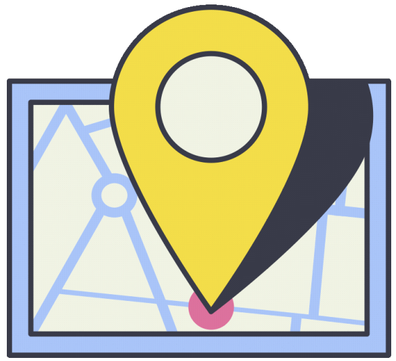


Technology Stack

Web Scraper

MySQL

ANGULAR



Java

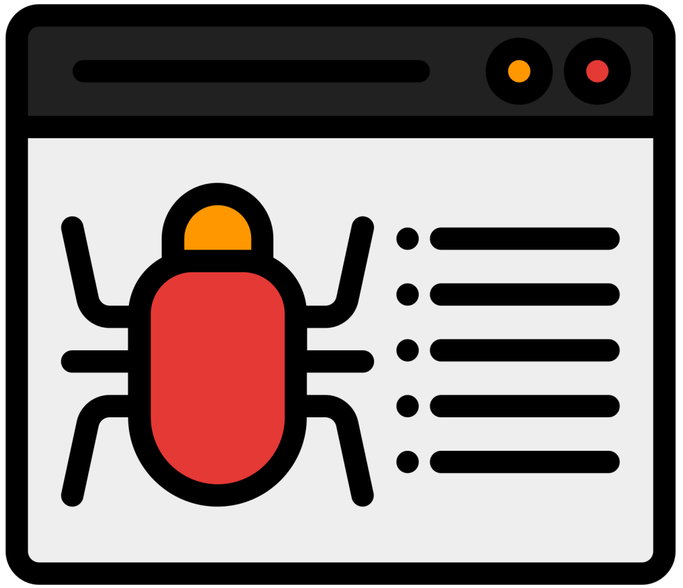
python™

spring

Crawler

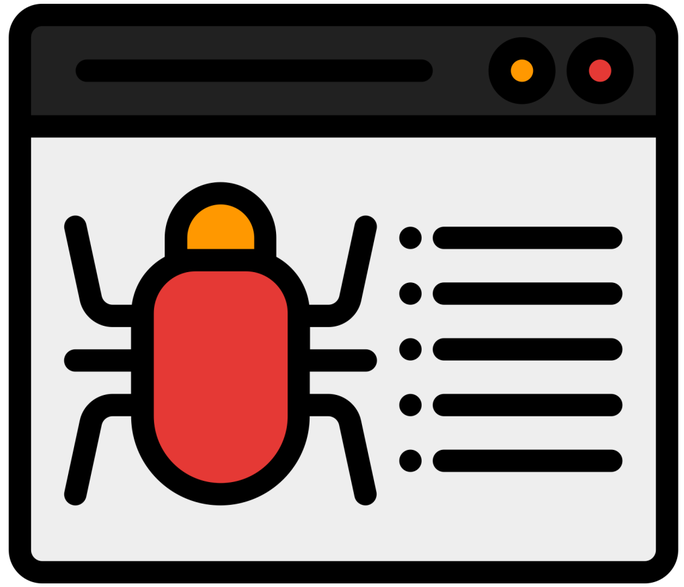
- We tried to use the crawler code for HW1 to Zillow but found Zillow has anti-bot measures.

- We have used Web Scraper(which will be located on Chrome-> View->Developer->Developer Tools)

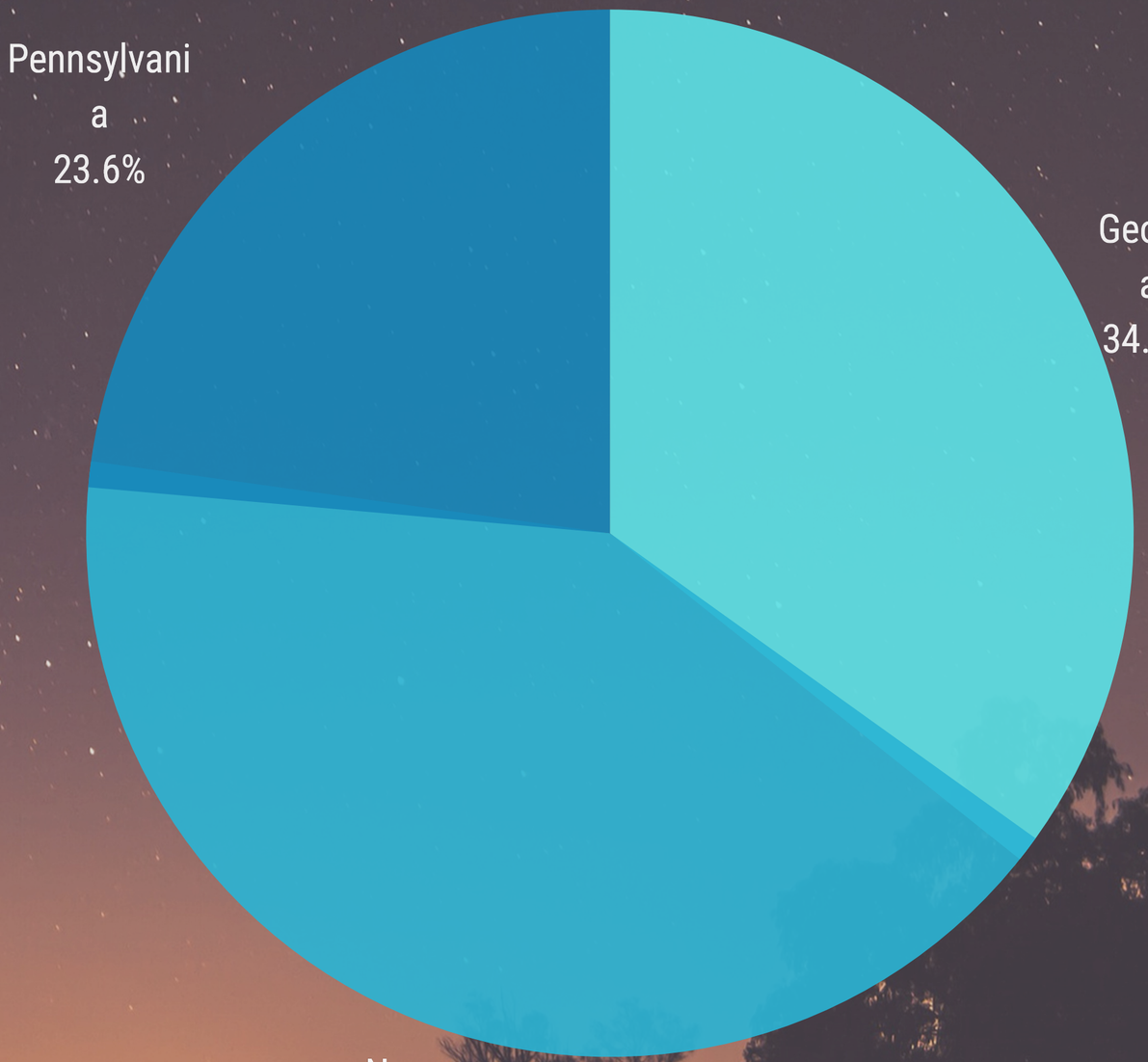


Difficulties Faced in Crawler

- We modified the code for HW1 to crawl zillow, but no data was obtained, which is caused by Zillow's anti-bot measures.
- Thus, we looked through other options in crawling and finally found out Web Scraper which uses IP rotation technique that uses a rotating proxy or VPN service to change the IP address from which the requests are sent. This technique makes it difficult for websites to track the scraper's activities and block the IP address.
- We learned that crawling websites is not difficult in terms of css selector, but it is when dealing websites with anti-bot measures.



DATA DISTRIBUTION



Backend Service

Java 1.8
Spring Boot 2.2.7
MySQL 8.0.30
Postman 10.12.0
Google Maps Distance Matrix API

Database Schema

Field	Type	Null	Key	Default	Ex
▶ id	bigint	NO	PRI	NULL	
address	varchar(255)	YES		NULL	
apartment_name	varchar(255)	YES		NULL	
images	varchar(1000)	YES		NULL	
latitude	double	YES		NULL	
link	varchar(255)	YES		NULL	
longitude	double	YES		NULL	
rent_1bd	bigint	YES		NULL	
rent_2bds	bigint	YES		NULL	
rent_3bds	bigint	YES		NULL	
rent_4bds	bigint	YES		NULL	
rent_5bds	bigint	YES		NULL	
rent_studio	bigint	YES		NULL	
scrapper_order	varchar(255)	YES		NULL	

Postman Endpoints:

- http://localhost:8080/api/excel/upload
- http://localhost:8080/api/googlemaps/addAddressCoordinates
- http://localhost:8080/api/googlemaps/fetchAvailableRentals

Search Algorithm

To find suitable housing options within the search radius entered by the user, we were originally using the following algorithm:

- Fetch geographical coordinates of the point of interest using Google Maps API

- Using the fetched latitude and longitude of the above point of interest, compute the latitude and longitude range for the DB query. This is done using a simple calculation where a 0.1 point difference in lat, long value corresponds to a land distance of 1 mile. Therefore for a search radius of **n** miles:

  start latitude = (latitude of the point of interest) - (0.1 * n),
  end latitude = (latitude of the point of interest) + (0.1 * n).
  (The same formulae apply to longitude.)

  This forms a square-ish search area instead of a circular one.

- Search for rows in the database whose coordinates lie within the range computed in Step 2.

Modified Search Algorithm

However, this algorithm was slightly incorrect and required modifications. We realized that simply relying on latitude and longitude gave us the direct distance between two locations. Therefore, if the user wanted to find a house within **n** miles of driving distance, using the previous algorithm was insufficient as streets may or may not be along the direct path between two locations. We added another component to the algorithm as follows:

- Use the Google Maps API to feed the commute type and find the commute distance between the housing option and the point of interest.
- If this distance is less than or equal to the user's search radius, include the options in the final shortlist of housing options.

This helped us return accurate results!

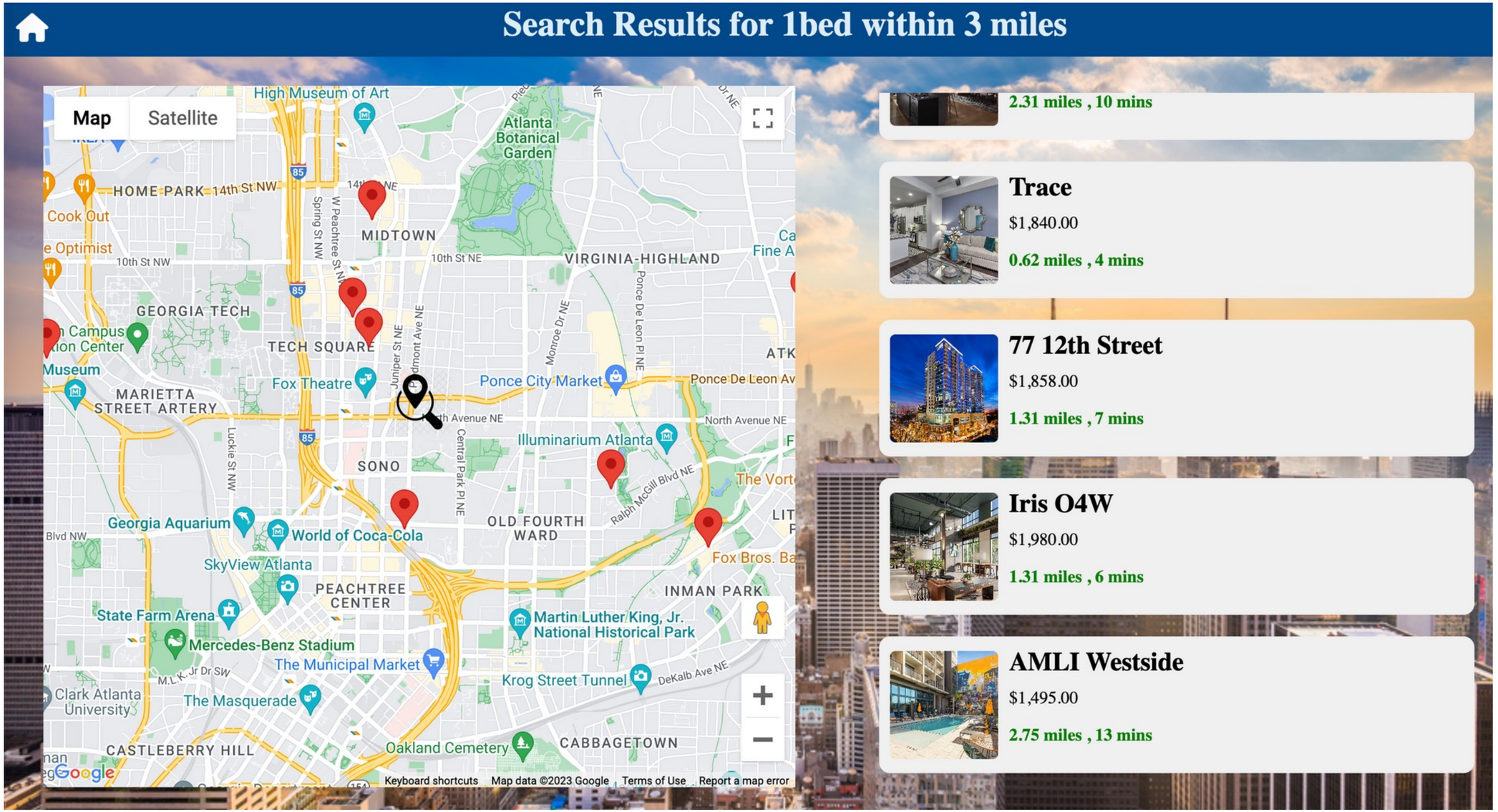
Custom options for User Search

User preferences input:

- Search Radius
- Commute Option
- Number of Rooms
- Points of Interest

User Preference Page (Landing Page)

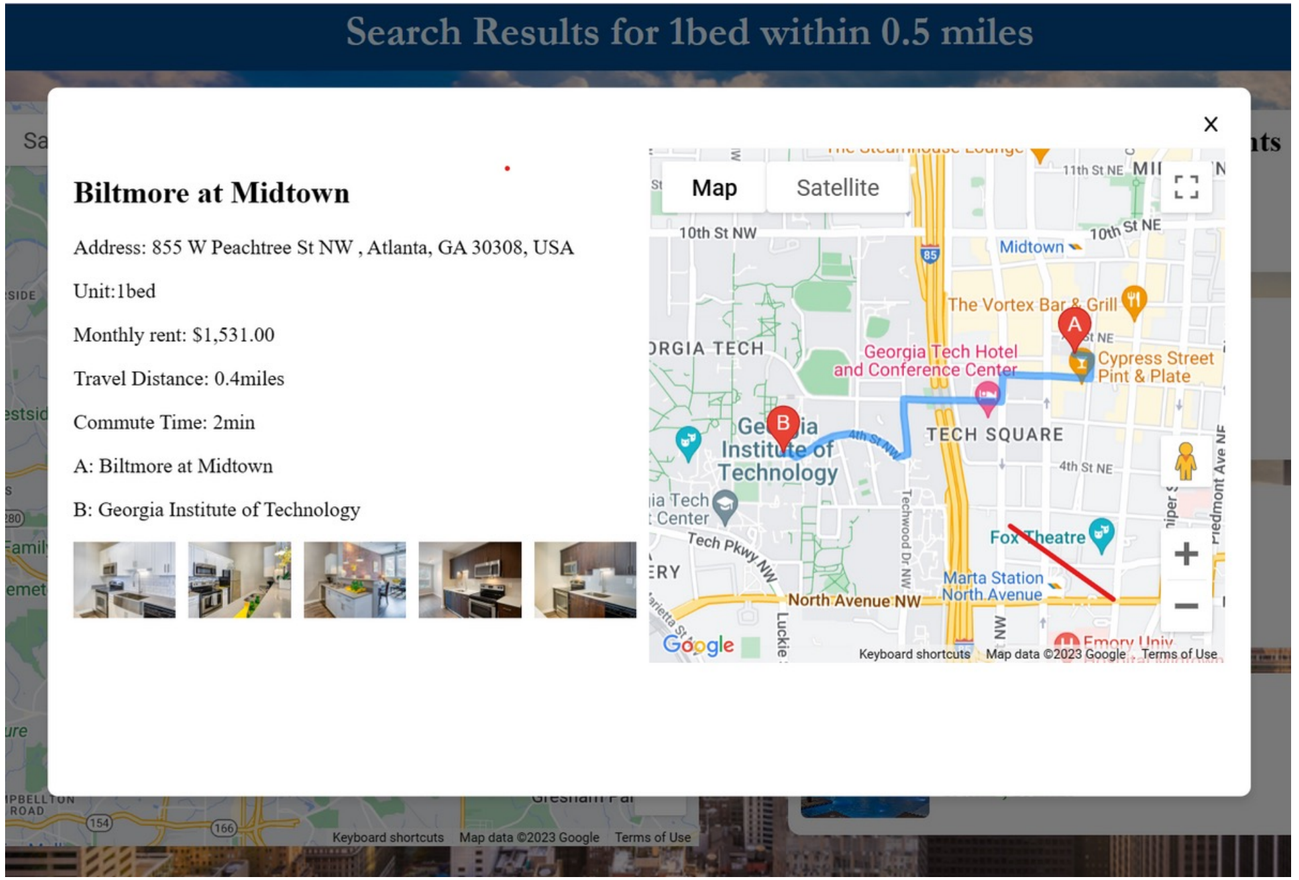
Search Results



Detailed Apartment View

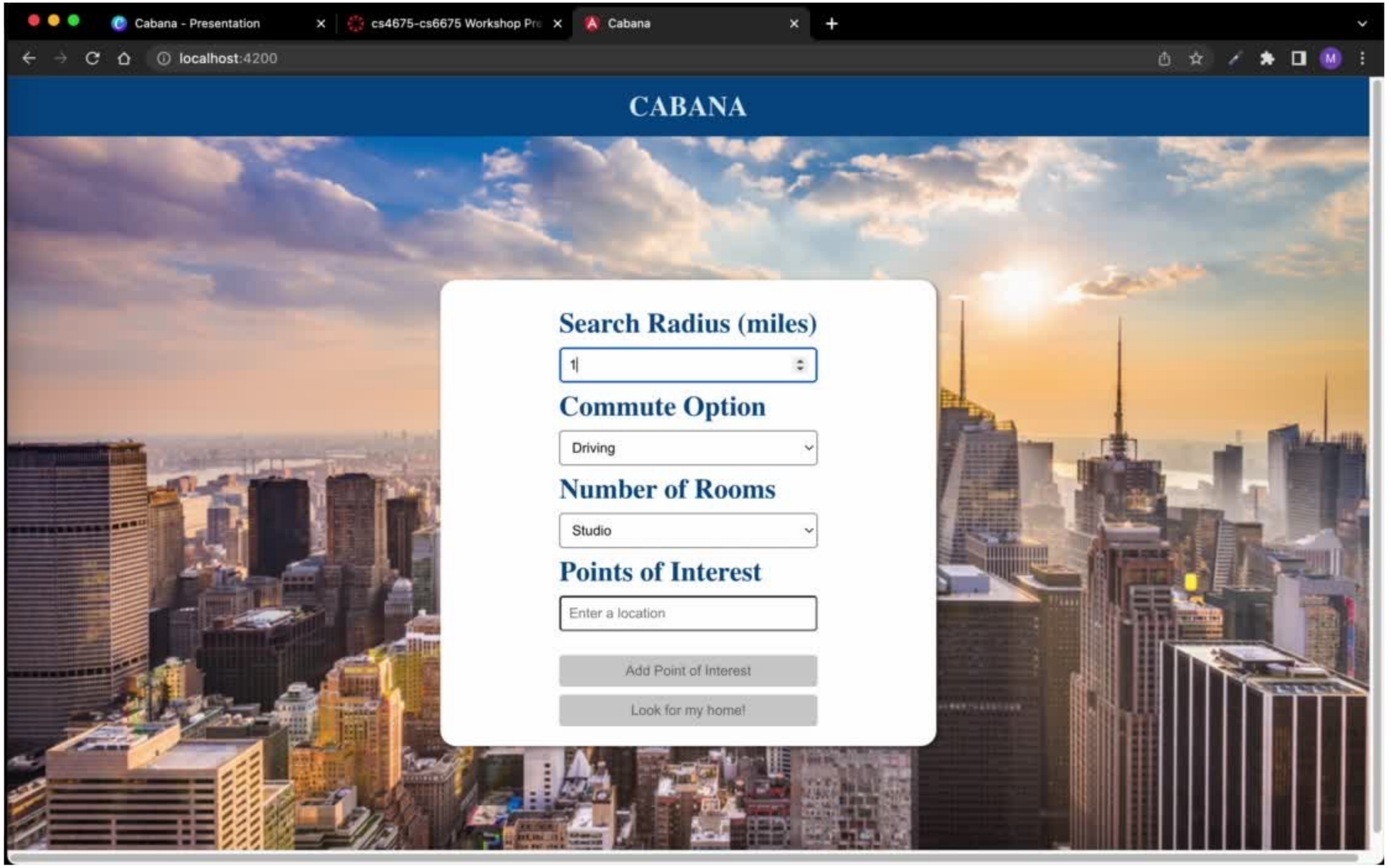
After receiving a response from the backend, we display a map with a location pin for each housing option. A list of cards where each card corresponds to one option, is also displayed.

On clicking on a desired card, we display another map with the *route between the user's point of interest and the option* marked out.



Demo Link:

Demo Video


Cabana - Presentation
cs4675-cs6675 Workshop Pre
Cabana
localhost:4200
CABANA
Search Radius (miles)
1
Commute Option
Driving
Number of Rooms
Studio
Points of Interest
Enter a location
Add Point of Interest
Look for my home!

Easy Case

- The user inputs one location and our software will display housings with the lowest commute time around that area from the database according to the user's commute choice.

Medium Case

- The user inputs one location and also can choose the mile range of housings displayed.
- Our software will display housings with the lowest commute time around that mile range from the database according to the user's commute choice.

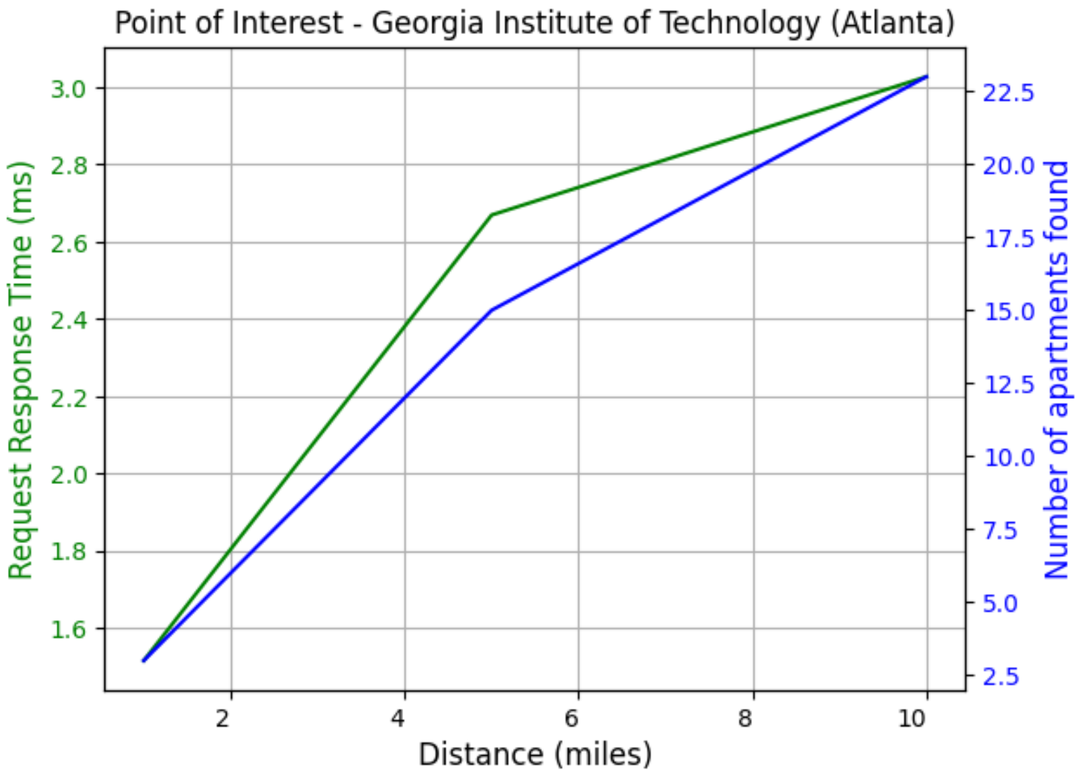
Hard Case

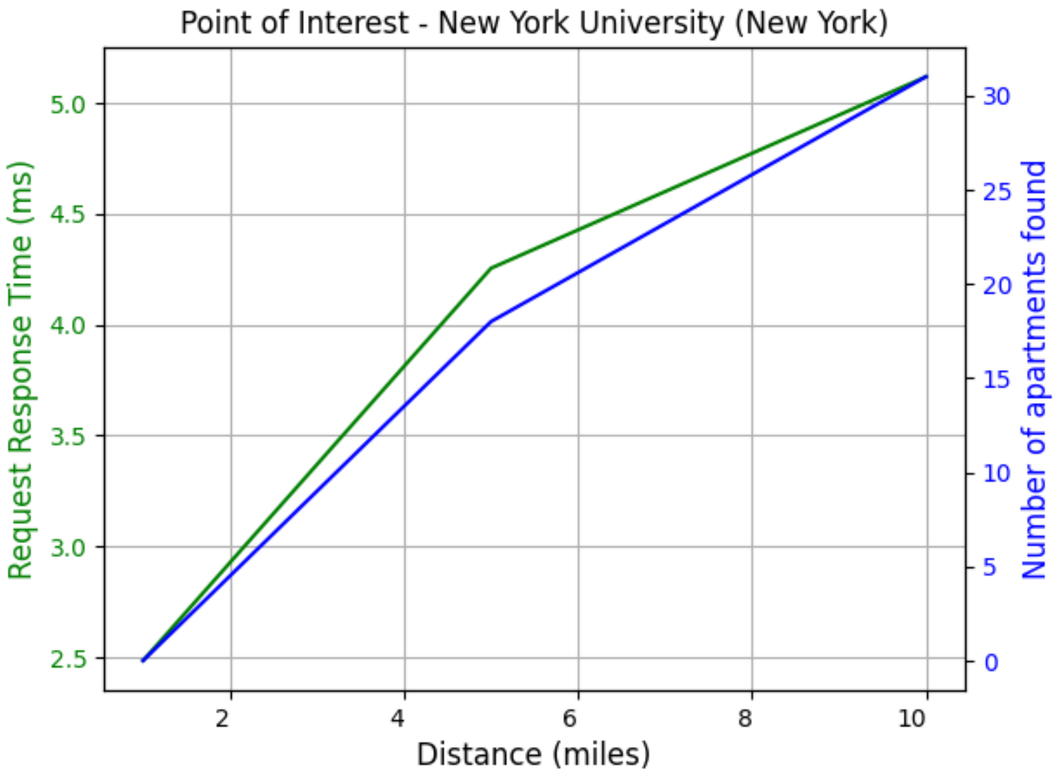
- Users can input several locations and our software will display the housings with the lowest total commute time for going through all locations.

Metrics/Datasets

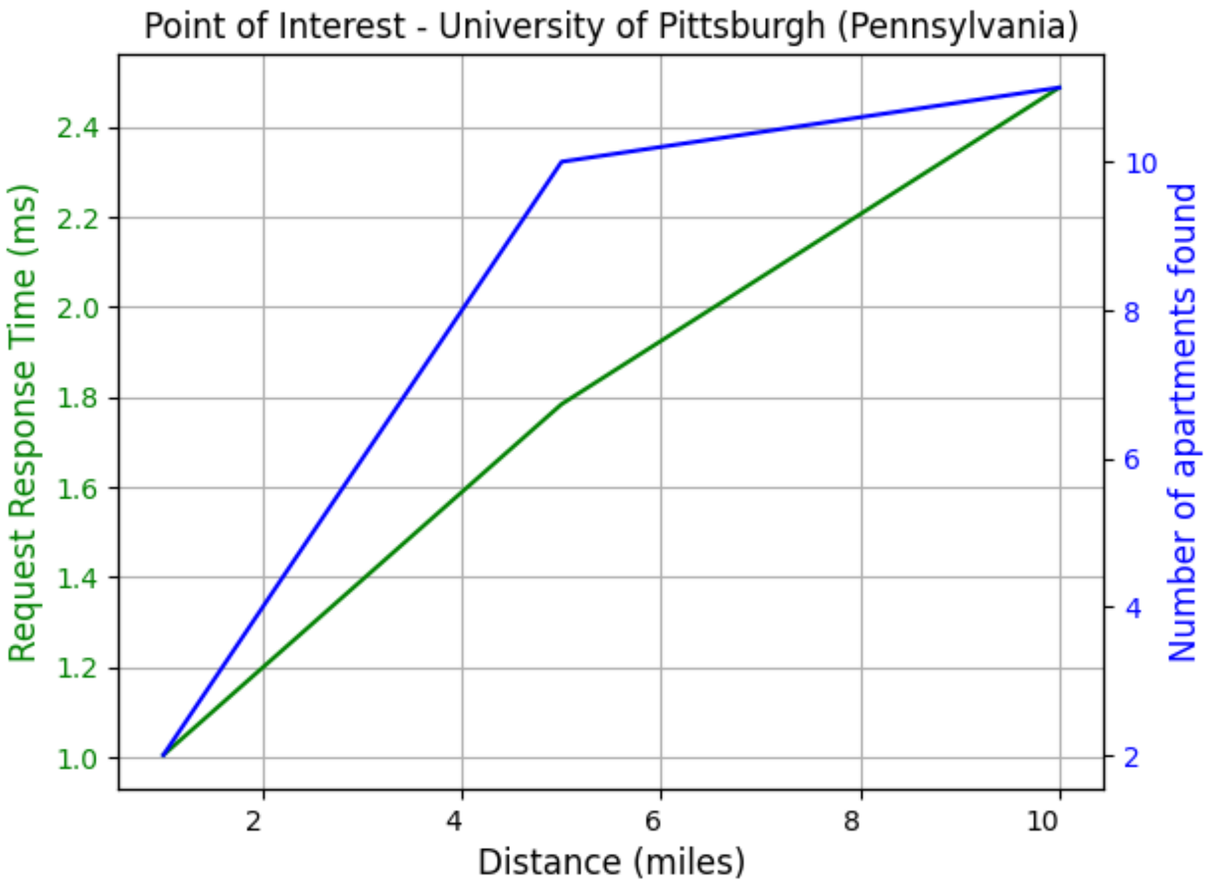
- As of today the database contains around 800 housing data.
- We cover rental properties across Atlanta, New York and Boston and plan to expand further.
- User will save 10% searching time using our website.
- It is more possible to find desirable apartment using our website compared to other websites since we integrate data from multiple sources.

Search Algorithm Performance
(Atlanta, New York, Pittsburgh)





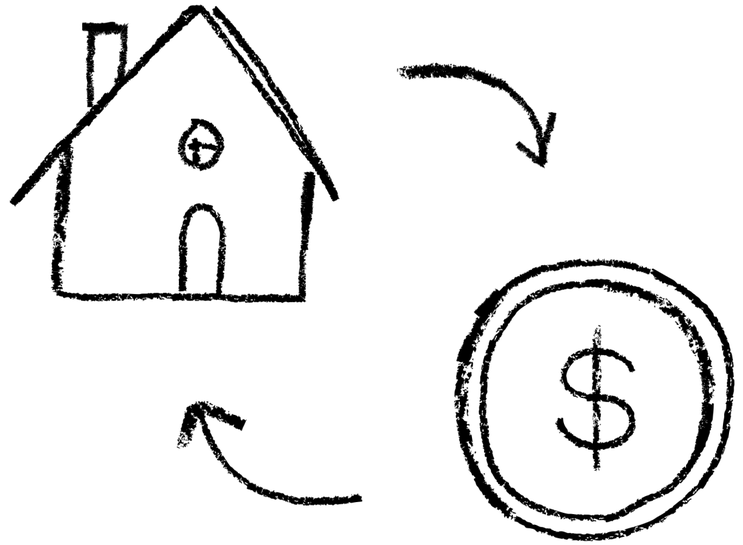
Search Algorithm Performance cont...
(Atlanta, New York, Pittsburgh)



The graphs illustrate that as the radius expands and more apartments are found, the request response time also increases.

How Cabana Adds Value (Strengths)

- Displays commute time from point of interest
- Gathers real estate data from various vast data sources
- Allows user to limit search to a particular radius
- Allows user to search based on preferred commute type
- Completely eliminates the user's dependency on Google Maps search
- As mentioned earlier, saves time!



Weaknesses/Future Enhancements

CRAWLER

Weakness:

- Our database is pre-crawled. (We are currently working on updating it once every day by a scheduled backend job that runs the crawler for new properties)
- Our pre-crawled data is limited to certain cities (Atlanta, New York, Boston).

Future Enhancement:

- We will implement synchronous crawling
- We will implement synchronous crawling with specific location entered by the user

Weaknesses/Future Enhancements

APPLICATION

Weakness:

- Designed to accept only one point of interest
- Currently in the process of deployment to cloud
- Adding additional user preference options such as price range and so on

Future Enhancement:

- We will expand the application so that a user can enter multiple points of interest through the UI and design an algorithm to give the most suitable housing options
- We will have a fully deployed website that can be accessed from any system with no additional dependency requirement