HomeValue

Matthew Brown, Mandy Wong, Kuang Sheng, Yanlin Song

Abstract—The potential user for this project is for people who want to buy their first house, but the information on the market are limited and questionable. In this project, the fittest home address will be provided based on the questionnaire which filled by user. The questionnaire is related with is the user's favor to certain facilities. By analysing the available house on the market, and all the searchable facility, including grocery stores, parks, etc. The team came up with an algorithm that can find few address that fit the user's requirement.

1 Introduction

HomeValue is a web application mainly focus on helping users find the best address of house that fits the values by which they live their lives.. There are three main categories which we considered based on trends in millennial value systems, but the application could easily be expanded to include many more criteria. The factors we considered were preference for healthy organic diets, preference for living with a low carbon footprint, and preference for exercise and outdoor activities. The way we decided to map these values to features in a home are, respectively, distance from the nearest Whole Foods, distance from the nearest major public transit hub, and distance from the nearest park.

One challenge that we say in this project was that we would need to aggregate data from multiple different sources. In order to keep the scope of the project small enough for this class, we decided to limit our scope to the San Francisco Bay Area for now. In the future, we could easily expand to other areas using the same techniques we did for the Bay Area.

The front end is a web application. It will take the user's answer and send it back to the backend program. The backend program need to use the data of all the selling houses on the market, the park location, transportation center information, and whole food market information. According to the user input, it needs to sort the currently available houses based on their similarity to the users ideal home.

2 Design thinking

2.1 Architecture Diagram

This shows our architecture diagram. The client accesses the website which is hosted in an AWS S3 bucket. They then specify their preferences through the UI which sends a request to our AWS Lambda function. The lambda function fetches the real estate data from an AWS RDS MYSQL instance and sorts the data based on the user's preferences. Then the UI displays the list of houses and plots them on a map.

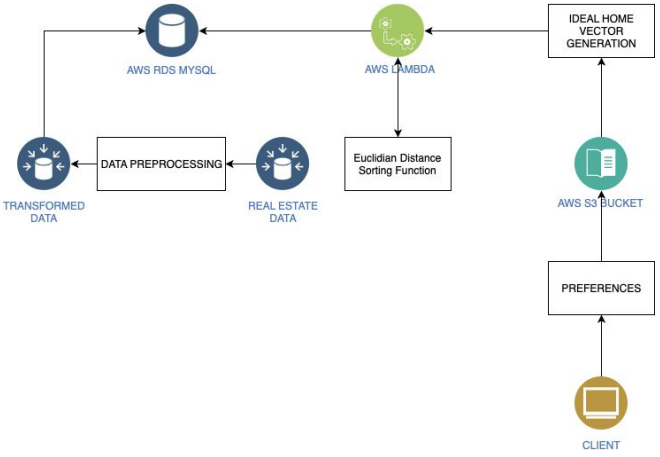

Fig. 1. Architecture Diagram

2.2 hills statement

A millenial looking to buy their first home can retrieve a list of houses for sale which align with their values.

2.3 user personas

HomeValue web application is targeted for the user who is in age of 25 to 34 years old with either a Bachelor's or a Master degree completed and currently working, would like to buy his/her first house.

The goal for the user to use HomeValue web application is to obtain house buying guide which best matches his family expectation. They can either do some research about the housing online or find real estate agent for suggestion. But, there are lots of housing information online and he/she will be spending so much time on just doing the research. Sometimes, it is really hard for them to find their "dream" house if they have so many requirements or preference for the housing.

The user can simply providing the list of the requirements or their expectations about their first house/dream house

including all the millennial trends and the web application will provide a sorted results with the preference the user provided so that user can save time for the environment of the housing check and can make their decision as soon as possible after the field visit.

3 Frontend web application

There are three pages in the web application. Login page, Questionnaire page and result page. The web application is basically built by HTML, CSS and Angular.

3.1 Login page

In the login page, we are using FirebaseUI Auth as the sign-in system of the application. It provides multiple popular sign-in options, including Google and Facebook login. It helps to secure the sensitive login information of the user. User could choose any sign-in option or just continue as a guest to use the service. Currently there is no difference between login user and guest user. In the future we are planning to save the filtered option for login user.

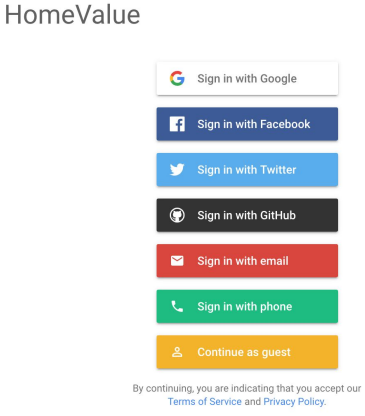

Fig. 2. Login Page

3.2 Questionnaire

The questionnaire is basically asking the user to fill in their preference of Park, Whole Food and Transit Station. After the user finish questions, it will generate a list contain the answer. After press the submit button, the list is sent to the backend and result is returned.

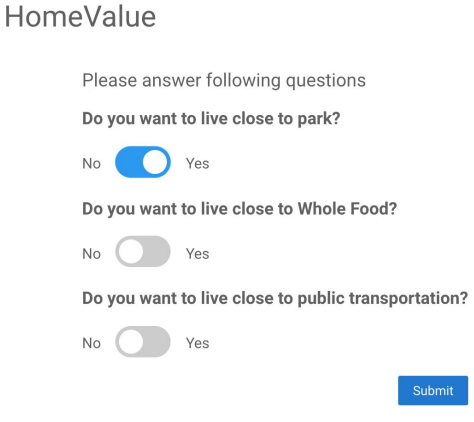

Fig. 3. Questionnaire Page

3.3 Dashboard

Dashboard is showing the sorted results based on the previous questionnaire. The map is at the left side and the result list is on the right side. The database of housing list is already converted into longitude and latitude in the backend database. Therefore, the result list can be shown on the map directectly.

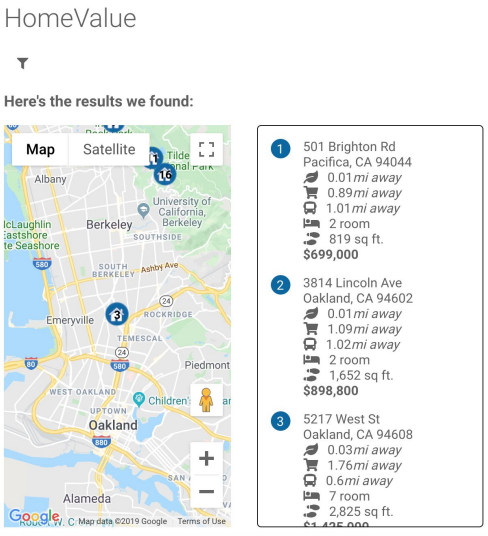

Fig. 4. Results Page

The details of home are showing in this order:

- street address
- city, state and zip code
- distance from the nearest park
- distance from nearby Whole Food
- distance from the nearest transit station
- number of rooms
- Sq. ft.
- home value

The user can still change the preference in dashboard page. there is filter logo on the top left right below the HomeValue. User can easily change their preference and the results will be refreshed after the user clicked the submit button.

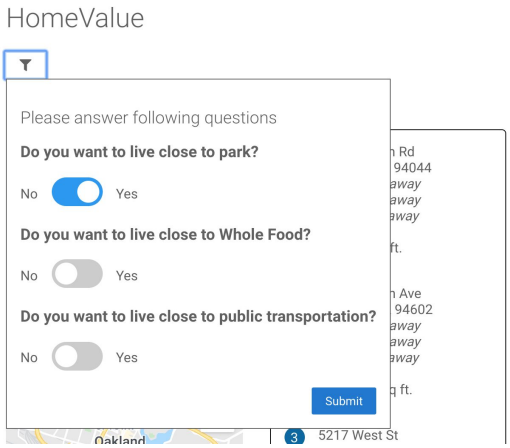

Fig. 5. filter in dashboard

3.4 Angular project

There are three major components in this angular project: top-bar, login page, and dashboard page. We are storing all questions into a constant file, which will be used in the dashboard page. Besides, we are also having 2 services: api service and map service. Api service is handling the logic call to server, which mainly focusing on retrieving results from server. For the map service, it is handling all logic related to the google map, such as initialize the map and add marker. Lastly, there is config which include all the sensitive info for the application, including firebase api and google map api keys.

4 Backend program

The first thing we did was obtain a list of houses for sale in the bay area. We did this by scraping real estate data from Zillow which included information like address, square footage, number of rooms, price, latitude, and longitude. Next we needed to do some preprocessing to figure out how far each house was from each of the types of locations we were interested in. While obtaining data for the project, latitudes and longitudes for parks, grocery stores, and public transportation hubs are collected by using google Geocode API. One Json file was returned by the Geocode API for one address. The location of latitude and longitude attitudes are located in the Json file. The latitude and longitude attitudes are stored in a pandas dataframe, and further stored in a csv file. The latitude and longitude for public transportation stations, whole foods stores, and community parks are also obtained and stored in a csv file. The distance between houses to each whole foods stores, public transportation stations, and community parks is calculated, and the shortest distance is stored in a dataframe and further stored in a csv file.

There are several csv files stored for the system. One csv contains the house information including house street address, price, latitude and longitude obtained by using google GeoAPI. One csv file for each of the following contents, address of the transportation stations which includes BART, CalTrain, and light rail in San Jose, address of the whole foods in bay area, and the community park address. Community parks are obtained from government websites. Public transportation stations address and the community parks address are stored as their names first and searched the latitude and longitude coordinates are obtained by using Google Geocode. Whole foods are stored as their street address, and the coordinates are searched by using Google Geocode. Three csv files were created including either a street address or the name of the location, and the latitudes and longitudes of the location.

The latitudes and longitudes of the locations are used to calculate the distance between two locations. The equation is presented as following:

distance=acos(sin(pi*lat1/180)*sin(pi*lat2/180)+cos(pi*lat1/180)*cos(pi*lat2/180)*cos(pi*long1/180 - pi*long2/180))*3963

Distance between each house and each Whole Foods location, public transport station, and community parks are calculated, and each distance between house and different places is put into a min heap data structure, and the minimum distance of each destination is obtained from the min heap. All the smallest distances are stored in a new dataframe with all house information preloaded in. The result csv file contains house information and three distances between house and the nearest whole foods market, the nearest transportation station and the nearest community park.

Our API was implemented using the Flask library and the Python programming language. The API consists of one endpoint, "/houses" which supports the GET method. It accepts three boolean query parameters called "checkParks", "checkTransit", and "checkStores". These boolean parameters correspond to the three questions that the user answers on the main dashboard of the application and setting them to true modifies the behavior of the application.

When the user answers the questions on the main dashboard, we use their responses to build a vector representing their ideal home. If they answer yes to all three questions, then the resulting vector is [0.01, 0.01, 0.01]. This represents a house that is 0.01 miles from each of the types of locations represented in the survey. We then compare this ideal vector to the actual vector representing each house and calculate the Euclidian distance between these two vectors. We then sort the houses according to their Euclidian distance from the users ideal home.

We deployed the backend on AWS Lambda which is a serverless computing product made by Amazon. Whenever the API is called, the server spins up and serves the request. It's only actually using compute resources while the request is being served, so the cost ends up being much less expensive than having an EC2 instance running all the time. The real estate data is hosted in an AWS RDS instance of a MYSQL database and the API queries the database when each call is made.

5 Conclusion

In conclusion, this project is mainly work for the people who wants to buy a house but don't know where to start. We feel that a house should compliment the lifestyle of the person buying it, and that looking for a house using that criteria makes much more sense than punching in specific zip codes to Zillow or Redfin. This project has great potential by adding more specific questions and larger database. The front end collected user answer, Then send the answer list to the backend. Display the list of house with google map. The backend gets the name or the street address of the place, and calculate the distance

between this house and the nearest public transportation station, whole foods, and community park.

The project is deployed on the AWS[1].

References

[1] http://cmpe272classdemobucket.s3-website-us-west-1.amazonaws.com/