**House Search Engine**

[**http://housing-site-host.s3-website-us-west-2.amazonaws.com/**](http://housing-site-host.s3-website-us-west-2.amazonaws.com/)

**Table of Contents**

Introduction2

Operation2

Searching Parameter Selection2

Searching Result Display2

New Listing Examples4

Code Implementation5

Front End5

Back End5

Future Improvement6

**Introduction**

House hunting can be very time consuming, especially at the “hot” area, e.g. the San Francisco Bay Area. Usually, people need to look through the popular Real Estate Websites, such as Redfin or Zillow. These website contains a lot of information about the houses on the market, but sometimes users will be overwhelmed by the huge amount of information and spend too much time looking through each houses.

The House Search Engine we designed is to help users to refine their searching and obtain the necessary information about the houses that match the searching criteria.

**Operation**

The UI of the House Search Engine contains three portions: (1) Searching Parameter Selection (fig. 1), (2) Searching Result Display (fig. 2) and (3) New Listing Examples (fig. 3).

1. **Searching Parameter Selection**

In the Searching Parameter Selection section, users need to input the below information:

* Street
* City
* Zip Code
* Property Type
* Price Range
* # of Bedroom
* # of Bathroom

The input will be used to compare with the database, which is the house information crawled from Redfin. Only the houses that match all the input criteria will be provided to users. If the parameters are left empty, these parameters will not be used in the searching.

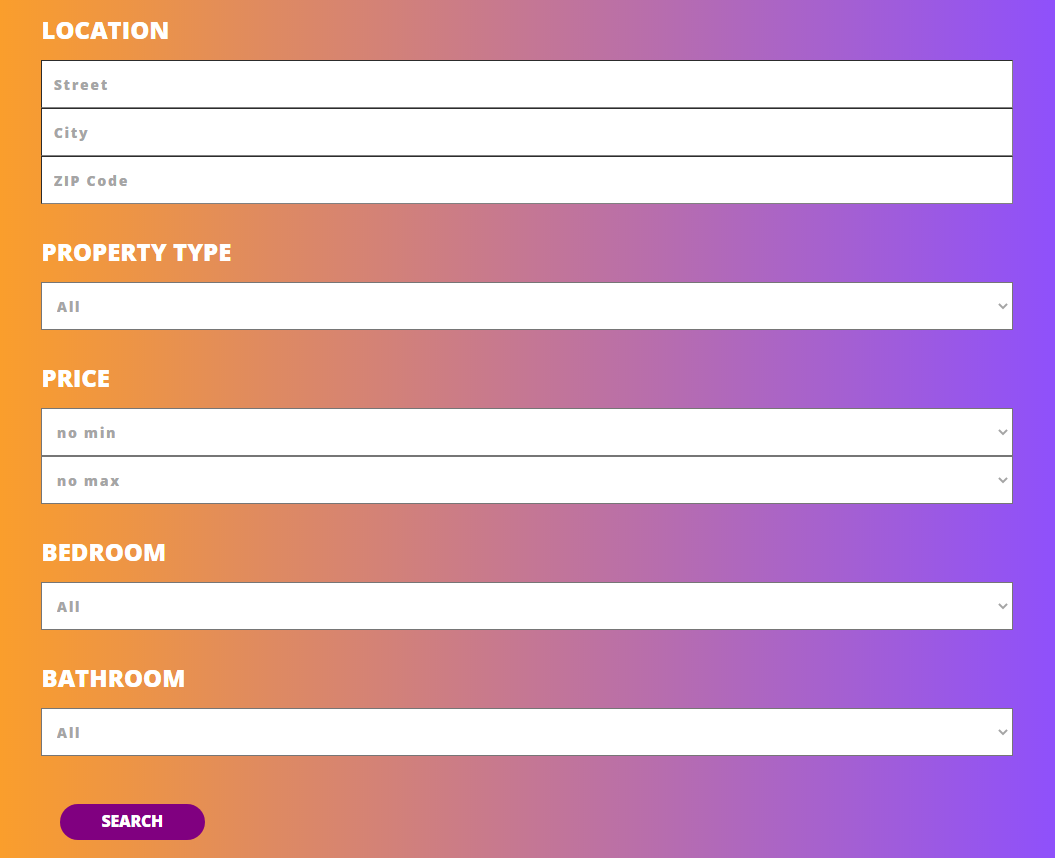
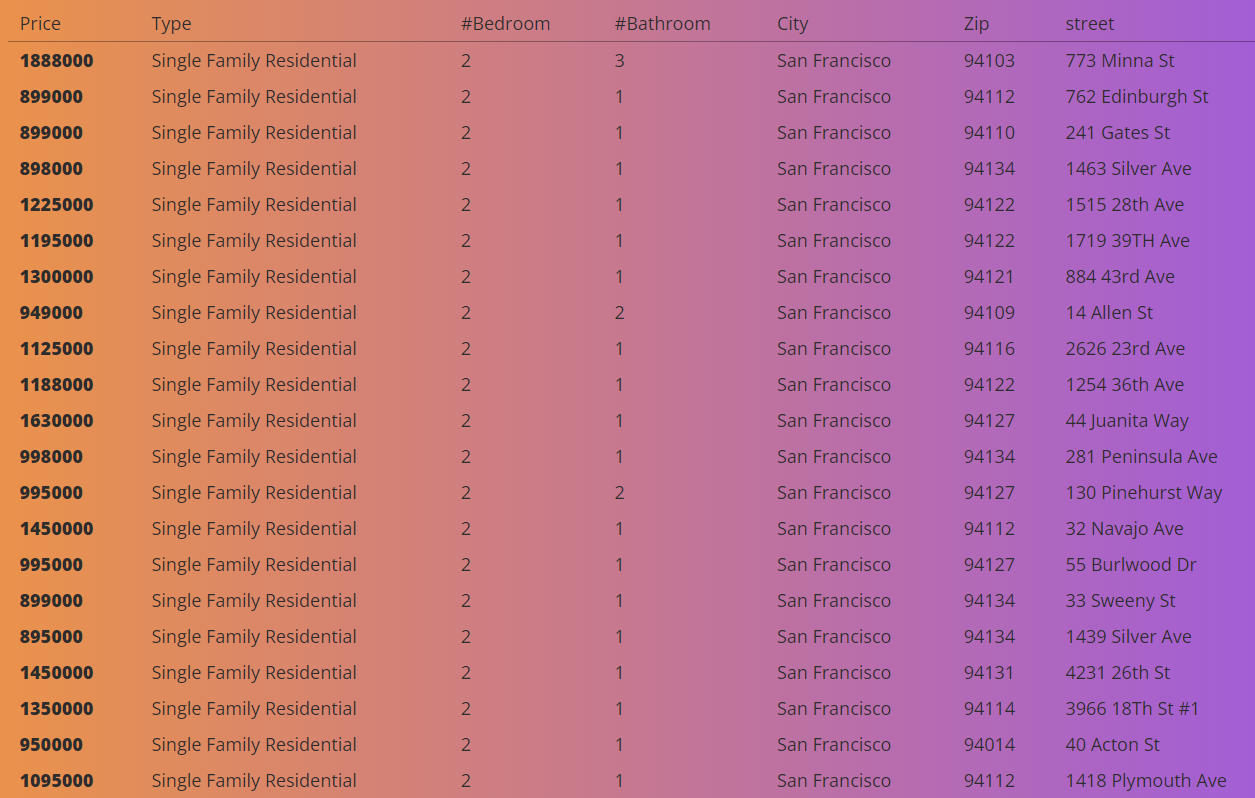


Fig. 1 Searching Parameter Selection section.

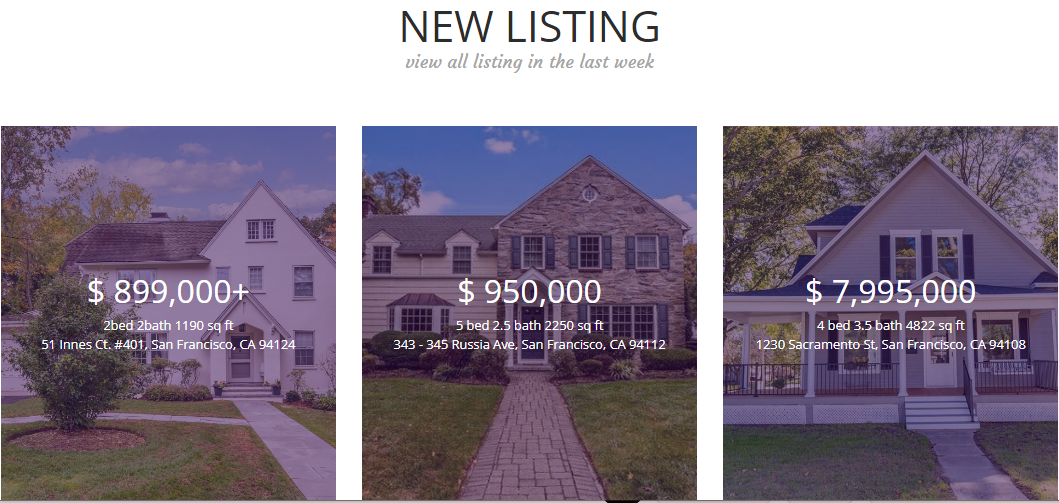
1. **Searching Result Display**

After users submit the search, the matched results will be displayed in the Searching Result Display section as a table. The table contains each parameter in one column.

Fig. 2 Searching Result Display section

1. **New Listing Example**

The last section is the New Listing Examples, which displays 3 examples of newly listed on Redfin. Currently, this section does not have extra functionality.

Fig. 3 New Listing Examples section

**Code Implementation**

1. **Front End – Xinyuan and Hao**

The front end website is created by React Javascript. The different sections on the websites are controlled by different files: Header.js, Main.js, Search.js, SearchResult.js and Newlisting.js (Fig. 4).

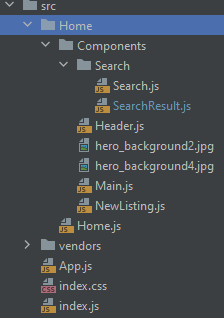


Fig. 4 Front End Code

Header.js displays the logo on the header

Main.js: displays the background image and the slogan

Search.js has three important functionalities:

* Collects the user inputs
* Perform the comparison with database
* Pass the searching result to SearchResult.js for displaying

SearcheResult.js gets the result from Search.js and displays the result as a table

Newlisting.js displays the examples of newly listed houses

1. **Back End - Hao**

A MySQL database was set up using AWS RDS to store the data. The table has eight fields: *id, prince, type, bed, bath, city, zip, street.*

A screenshot of a computer

Description automatically generated with medium confidence

These data are converted using a node server to JSON format. A screenshot of a computer

Description automatically generated with medium confidence

Initially, we are going to use this as an API server. But we found storing the JSON file in an AWS s3 bucket and serve it is more efficient. So we put the converted data in AWS s3 buckets and served it from there.

The front end is able to fetch the JSON file directly from the s3 bucket.

1. **Data collection - James**

The data was collecting using a python script to scrape the data from the redfin website. The script requires a version of python that supports the BeautifulSoup package. This can be done by running a Python 3.7 environment using Anaconda.

The script then pulls a list of urls from Redfin. When searching up houses on redfin, it is possible to download a csv list of houses that contain their redfin url with the Download All button. In this case, we downloaded all of the housing data from the San Francisco area.

The scraper also needs the latest version of chromedriver, which can be downloaded from <https://chromedriver.chromium.org/downloads>. Once chromedriver and the list of urls has been placed in the same folder as the scraper, the script is run in a terminal with python.

The script goes to each url listed in the downloaded csv and looks at specific locations in the html of the visited page to gather numerical and text data related to the house. This data is then saved to output files.

This generates 2 files. One file is called textData.txt, which contains all the text data related to a house on redfin, along with its url. The other is called sqldatabase.csv, which contains numerical data from the house and is used to populate the database.

Text

Description automatically generated  
Fig. 5 Downloaded file and chromedriver in the script

**Future Improvement**

The current version has not fulfilled all our initial design due to the limited the time. For example, the original plan is to add weight on different searching parameters to further refine the searching. The future improvements and how to implement them are proposed in the below list:

1. Add weight to each parameter:

Users can rate each parameter (1~5), and the weight will be calculated by **rate/total rate.**

Ex: Price rate = 3, # of bathroom rate = 2 and # of bedroom rate = 2. The weight will be Price weight = 3/(3+2+2), # of bathroom weight = 2/(3+2+2) and # of bedroom weight = 2/ (3+2+2).

1. Add more house features:

More parameters can be added, such as house size or year built.

1. Result displayed with url:

Current result is displayed as a table with all the parameters. The url will be added in the result, so that users can directly link to the Redfin website to check more details about the interested houses.