

**To run:**

1. Open terminal and locate directory of the app.py file (using “cd”)
2. Enter “python app.py” to start the app
3. Go to: <http://localhost:5000/> on chrome to open the webpage

**PocketProperty** is a home address search that returns matching houses from a large set of houses from RedFin. User can provide search input and see results. Or they can scroll through the listing.

**There are two buttons:**

1. One provide details on the home (i.e. price, location, square feet, etc.)
2. The other redirects the user to the RedFin page for images.

**Data**: The data was extracted from <https://www.redfin.com/city/17151/CA/San-Francisco>

In the set, there are 350 home. The dataset is found in the house.csv file.

**Technology Used:**

Front End - HTML/CSS (some Bootstrap)/JS

Back End – Python (Pandas) + Flask

**Design Process:**

I had built simple website connected to a database using Python and Flask before. So I decided reusing the same technology will be the best place to start. I could’ve read up on Node/Rails but given the time constraint, it would be a difficult challenge to finish the project. I have never used JS in my website so that was something I had to read up on. I watched some videos to understand the idea behind DOM.

I could see the project was broken up into 3 components:

1. Connect and parse through the data on the backend and then to the front end
2. A way of filtering results using 1 endpoint (address)
3. Returning the right data given the select endpoint. Then show it on the website.

There were a few challenges along the way. Parsing the data in the csv and passing it to JS proved challenging because of the way the data was structured. I remember JSON works well with JS so I used pandas to convert the csv into a dictionary.

I looked around for a few ways to implement a search address and came across a filter list. I thought it would work really well for this project because you can display all the address and filter through them. The UI was great for searching. I copied the code from W3 School and played around with it.

At first, I only used 5 data points from “address” and “price” because it was easier to test and implement.

The hardest and most time consuming aspect was matching the address with the correct information. And figuring out how I would present that information.

I came up with a few ideas, and they were probably possible to implement, but I currently don’t have the background to execute. For instance, I was thinking I could create a new class id for each new detail (i.e. price) and then I could match the id with the address to present the information. I also tried using an alert window to present the information but I couldn’t pass information incrementally through it (only the first data was recognized). Finally, I found a button solution that could present information incrementally. But the issue was that after filtering the address, the buttons would still remain on the page cluttering it up. It might’ve been possible to create a JS function to hide it under certain condition, but I couldn’t execute.

After going back and forth, I found instead I could create a filter table instead of a list. The information was actually on W3 School so I played with the code. And got a corresponding button to display information for the right address!

Once I got 5 addresses working, it was easy to implement it for the 345 other ones. After everything was working, I played around with the styling to make it look less “boring”.

Overall: It was a lot of trial and error (creating separate files to test different ideas), taking breaks in between to come up with more optimal approaches, before getting to my solution.

**Further Improvement:**

Ability to search other data in the address bar.

Format data (i.e. price)