O’Hoon Kwon

Redfin Application Write-Up

Interacting with the API is quite straightforward and the majority of the complexity would come from creating a user interface that translates into customizable queries that interact with the API.

The application interface would be a simple one page application with a table. Each column would include an input for the user to adjust the parameters of the query made in the API request, such as: filtering by a specific day of the week, or ordering by the business hours. It would be optimal for the engineer to build the table using a library such as bootstrap. A table is the most straight-forward and understandable format to present the data to the user, as well as framing the adjustable parameters that the user would interact with.

The back end would involve translating each of the adjustable inputs into a string, forming the URL parameters that call the API. It would need to concisely pull together the query in the right order, such as putting all the WHERE parameters in the same section of the string. Once the backend has parsed and formatted the variables, we will address performance. When taking into consideration a potential large user base, calling the API repeatedly in large volume has many demerits, such as incurring a financial cost per call and the potential for overloading the API. To prevent this, memoization would be implemented, storing the data in our own database so that we make a call to our own database before the API if the criteria for the query has already been stored previously. Otherwise, we make the call to the API and store the data for future use. This data should exist on a timer and have data wiped when a certain amount of time passes, ie. a month, to account for API data changing.

Regarding the structure of the local database, should we choose to make this memoization more robust, would involve storing each part of the data as individual rows, essentially recreating the API’s data in our own database. This would be ideal in that it avoids writing repeat data, but introduces the possibility of potential gaps in our data. This approach would require more development time and a more thought out database architecture. Thus, if this is outside of the scale of the project, then we would defer to storing each call as a single json, pointed to by the combination of user input filters.