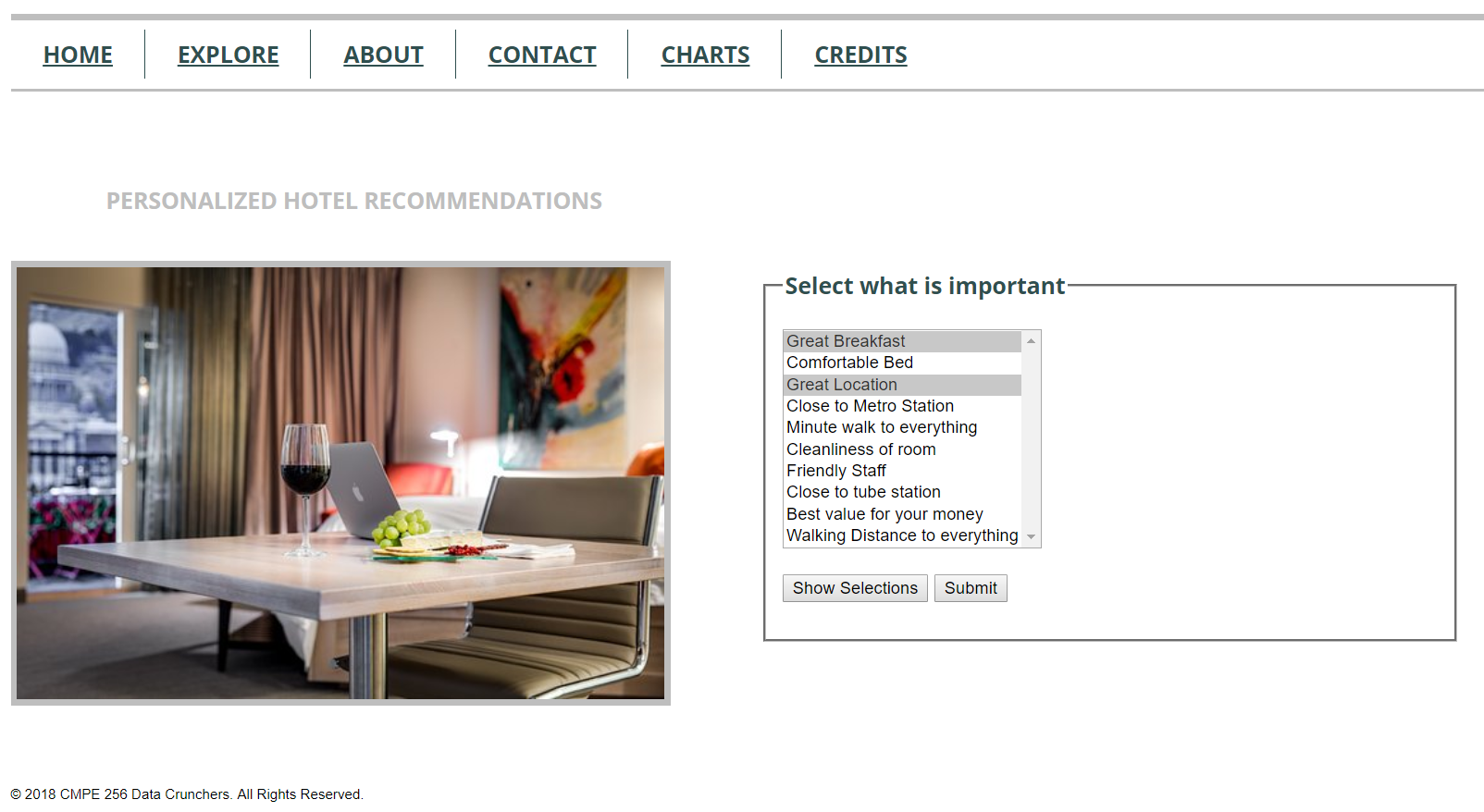
# Appendix 1

This appendix includes a detailed description with code snippets of how the recommendation application works. Full code is located in our GitHub repository.

## Implementation

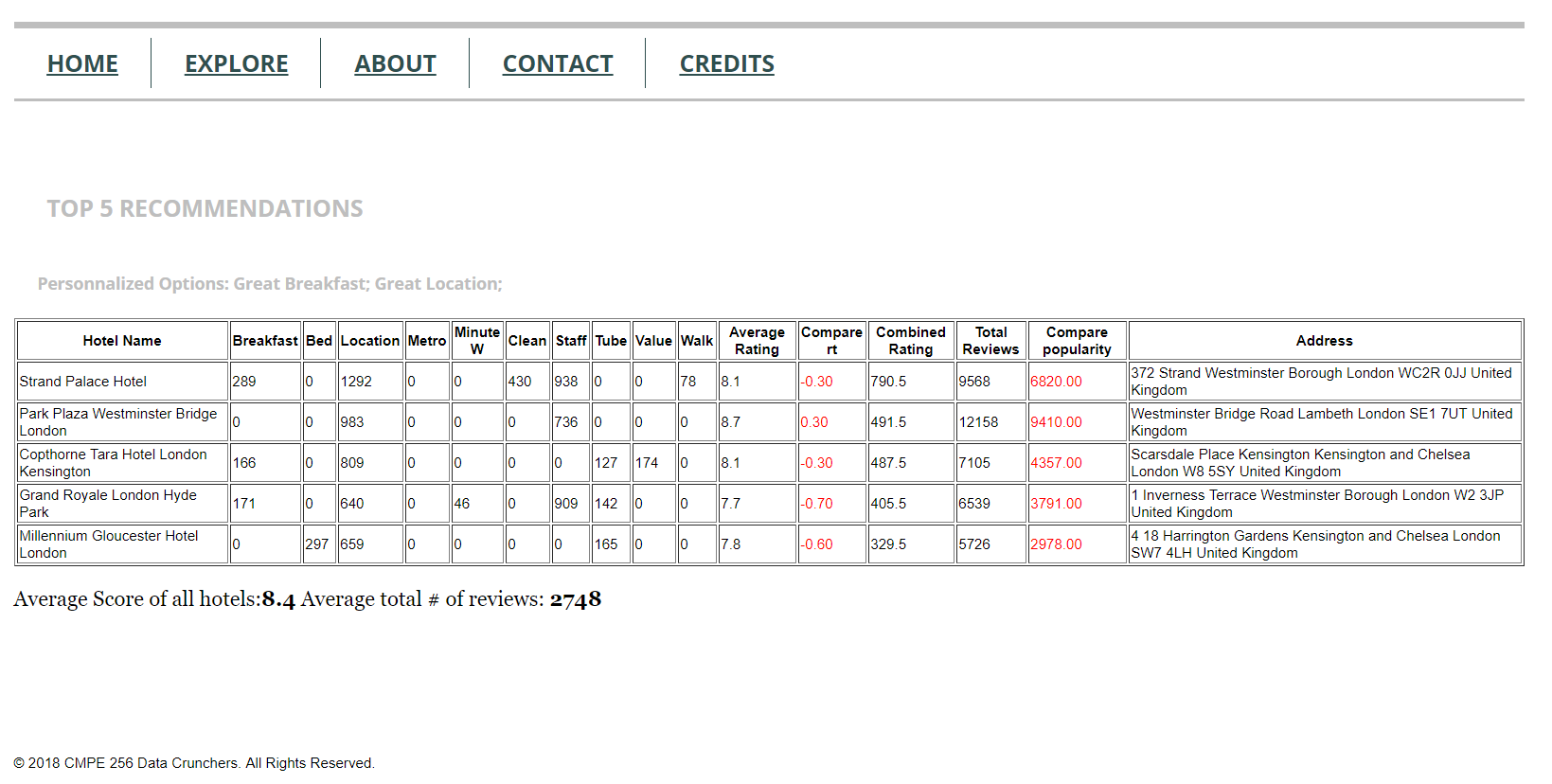
The user selects the categories that is important to them personally. As the report states, these categories are the result of the top 10 bigrams extracted from the positive review column.

Let’s say a hotel seeker only cares about Great Breakfast and Great Location. The hotel seeker would then select these two categories as shown below:



The next step is to click Submit. (The Show Selections button is shown for testing purposes). When you click submit, our algorithm will calculate the combined score of the two categories divided by the number of chosen categories. The data is extracted from Mongo database.

Below screen shows the result from these two categories:



For testing purposes, we show the result of each category. However, for this result, the only relevant scores are Breakfast and Location. The column that shows the combined rating, is the Combined Rating column. Let’s walk thru an example for how to derive the combined Rating. It is calculated by taking the score of breakfast (209) + score of Location (292) divided by two which is calculated to be 790.5. This is done for all hotels. Strand Palace Hotel has the highest combined score so this hotel will appear on top for these two selected categories.

In short, our algorithm will summarize and divide according to how many categories selected. The below section will explain the code.

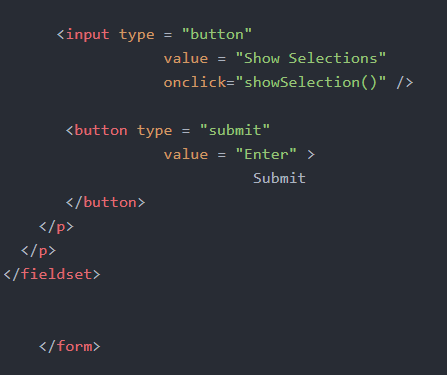
We start out with an html form = form1.

We have 10 unhidden input types + 10 visible variables that the user can select from in a menu selection. The user can pick any combination of values.

The unhidden input type stores is initially defaulted to 0.



There are two buttons in the form. One button is to showSelection (mostly for testing purposes) and the other button is to submit to get recommended hotels.



Two things happen when you click the submit button



1. Function ShowCategories()

Below is a code snippet of this function.



It detects which category the user selected and sets the value to 1 if clicked. Remember that

the category values were initially set to 0.

2) On submit user is redirected to url /get\_recommendation

We are using the express framework with node.js so when the submit happens, the program first checks what code to execute.

The route file index.js handles this part.



When the app detects html post and ‘/get\_recommendation’ then code modelMain.post\_recommendation is executed.

The below code snippet shows that we are using Mongo database.

We loaded all the hotel recommendations into a collection in Mongodb. Then we select all the recommendations and load them into an array called hotels.

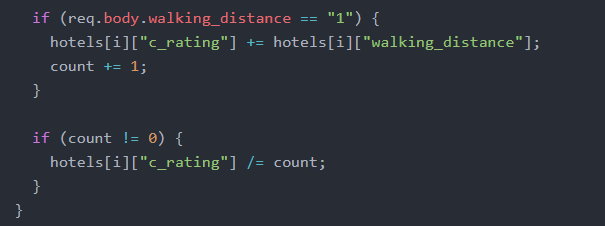


We then loop through the Hotels array as shown below. We only show a snippet of the three first categories.



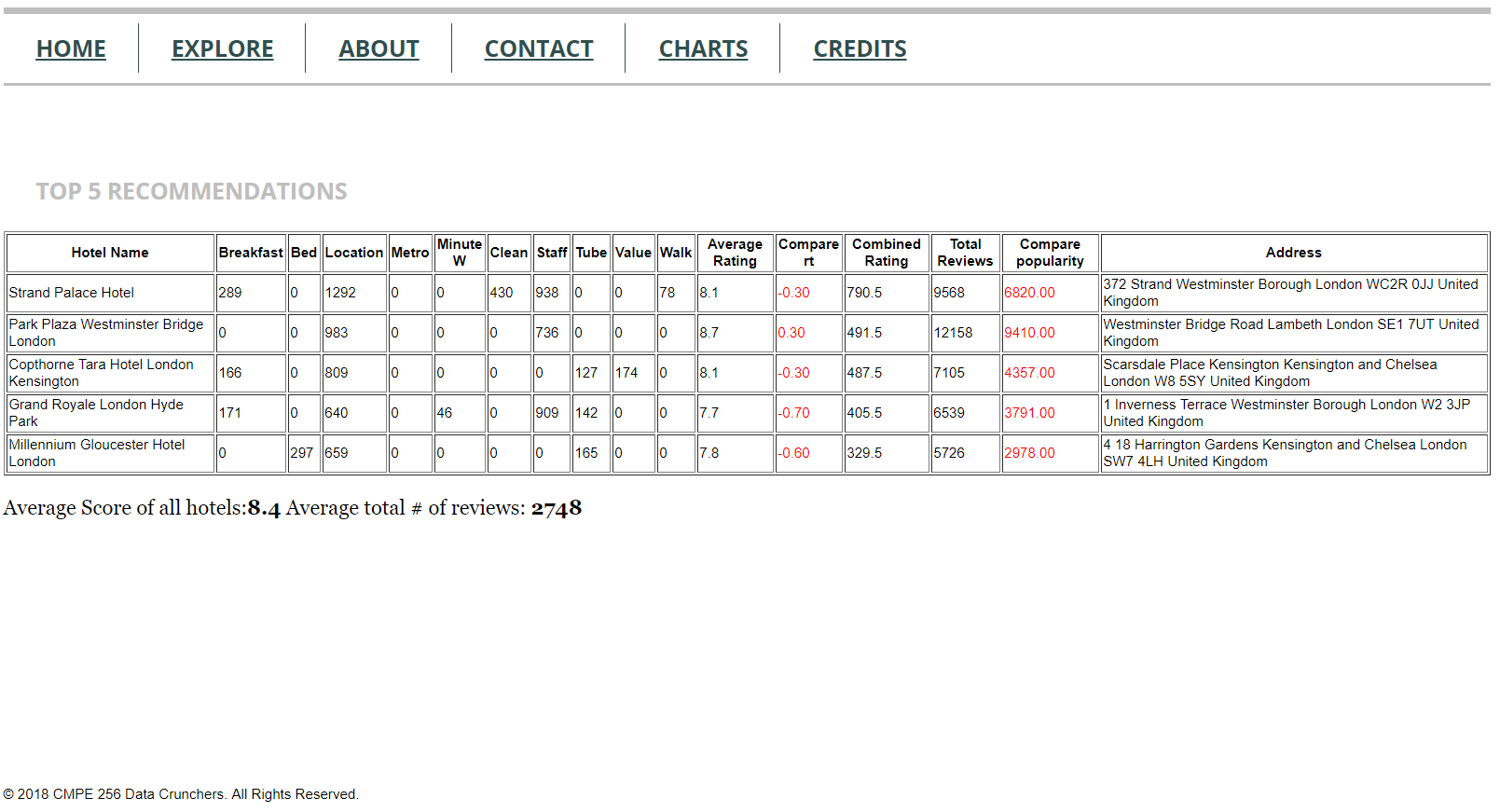
Notice that we also create two additional columns so that we can evaluate our algorithm against overall average rating of all hotels and overall popularity of all hotels.

Lastly we take the sum of all scores for the categories selected and divide by the # of categories selected as shown in below code snippet.



Then we return the top 5 hotels with the highest combined score in descending order and redirect the result to the recommendation page for the user to see.





## Evaluation

This model can be evaluated using A/B tests. Our approach to find the top 5 hotels for a user is a very personalized approach. It is tailored to what the hotel seeker values. However, our model evaluates pretty well. For randomly picked combination of categories, the top hotels is well above the average popularity. Remember that # of total reviews represents popularity. This result is expected because we would think that the hotel that has the most positive words also has many reviews. In terms of average rating for the hotel, the result is ok. At least, for every combination, the result is around the average or higher.