**Narrative for Final Project**

***The project url in my github is https://github.com/sharycherry/mcdonalds.***

My final project is intended to provide McDonald’s food recommendations to users. I broke it into two parts, one is an efficient crawler to extract food nutrition information from McDonald’s website. The other is a web-based recommendation system interface to provide personalized diets according to different requirements.

The crawling process is divided into two main parts.

First, I manually explored how human find the nutrition information step by step. So I went to the McDonald’s homepage https://www.mcdonalds.com/us/en-us.html, and found that this homepage includes a heading “OUR MENU” that list all categories of food, including Breakfast, Burgers, Chicken & Sandwiches, Salads, Snacks & Sides, etc. When I click one of the categories, it will go to the corresponding items page. For example, if I click the “Burgers” category, it will direct to a page showing all burger items includes “Big Mac”, “Double Quarter Pounder with Cheese”, etc. Each item contains a “LEARN MORE” hyperlink. After clicking it, it goes to a page showing the details of the item, like nutrition summary. When clicking the “VIEW NUTRITION SUMMARY”, it shows all nutrition information I want.

The second step is to simulate the above process by Python code. Basically, when we click an icon in the web page, the browser sends HTTP request to the URL behind that icon. So we have to find the URL in the HTML source code of the web page. And I chose to use Python beautifulsoup package to locate the URL. For instance, we need to get the URL of each category in the HTML source code. I first searched some key words like “Breakfast” in the source code and found that all the menu category URLs are in the div element “menulist-flyout”. So I used beautifulsoup to retrieve this div element part. The “menulist-flyout” div element contains many li tags and each li tag includes one menu category URL. Finally, I used Python requests module to send HTTP request to all the URLs.



Then, I repeated this way to get all the item page URLs. However, when coming to the item page (e.g. Big Mac), our traditional way fails because the HTML source code does not contain the item’s nutrition information. So how does the page show the nutrition information? I used the Chrome DevTools Network panel to analyze all the requests that come in and found that there is a request called “<https://www.mcdonalds.com/wws/json/getItemDetails.htm?country=US&language=en&showLiveData=true&item=200463>” whose response is actually the nutrition information in the JSON format. And it seems that the browser finally shows the response using Javascript. Therefore we can use Python code to send the same request to get the nutrition information. But how can we get the request URL of each item?

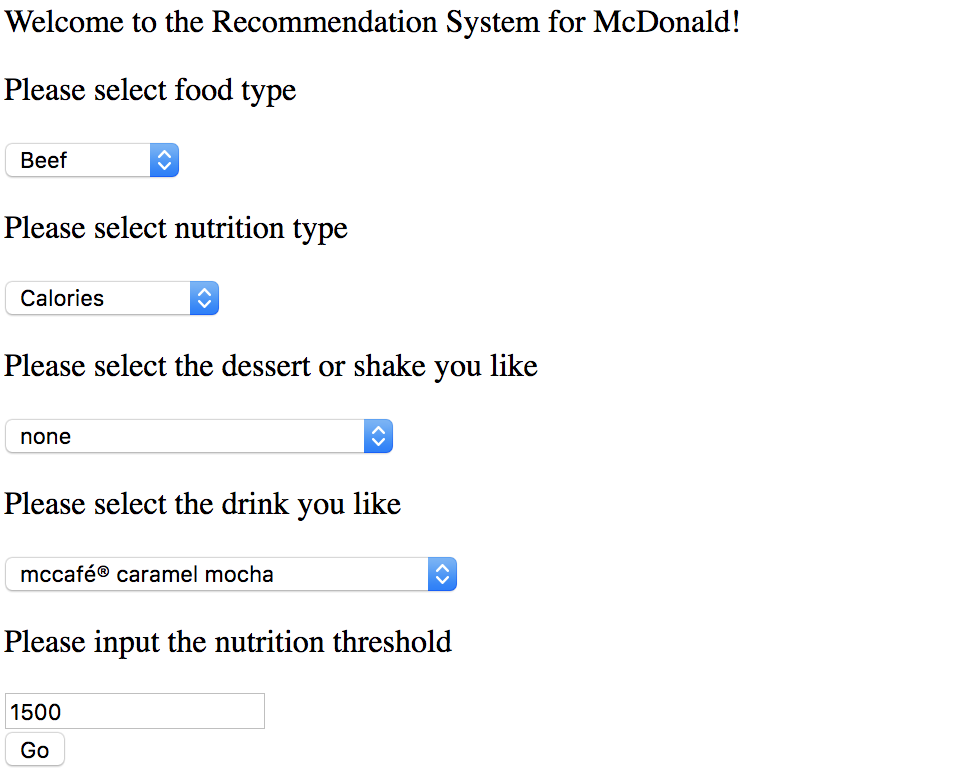
I started to check another item page like “Quarter Pounder with Cheese”, it sends a similar request “<https://www.mcdonalds.com/wws/json/getItemDetails.htm?country=US&language=en&showLiveData=true&item=200466>". And the difference between the two items is the end of the request. One is “item=200463”, the other is “item=200466”. So the request URL is actually the base URL “https://www.mcdonalds.com/wws/json/getItemDetails.htm?country=US&language=en&showLiveData=true&item=” plus the item ID. I guessed that the HTML source code may contain the item ID. Luckily, it is true. And the remaining part is the same as what I already did in the traditional way.

Finally, after getting the request URL of each item, I used Python requests module to get the response, which is the nutrition information I want. I also use Python code “db.py” to save the data into a database called “data.db”.

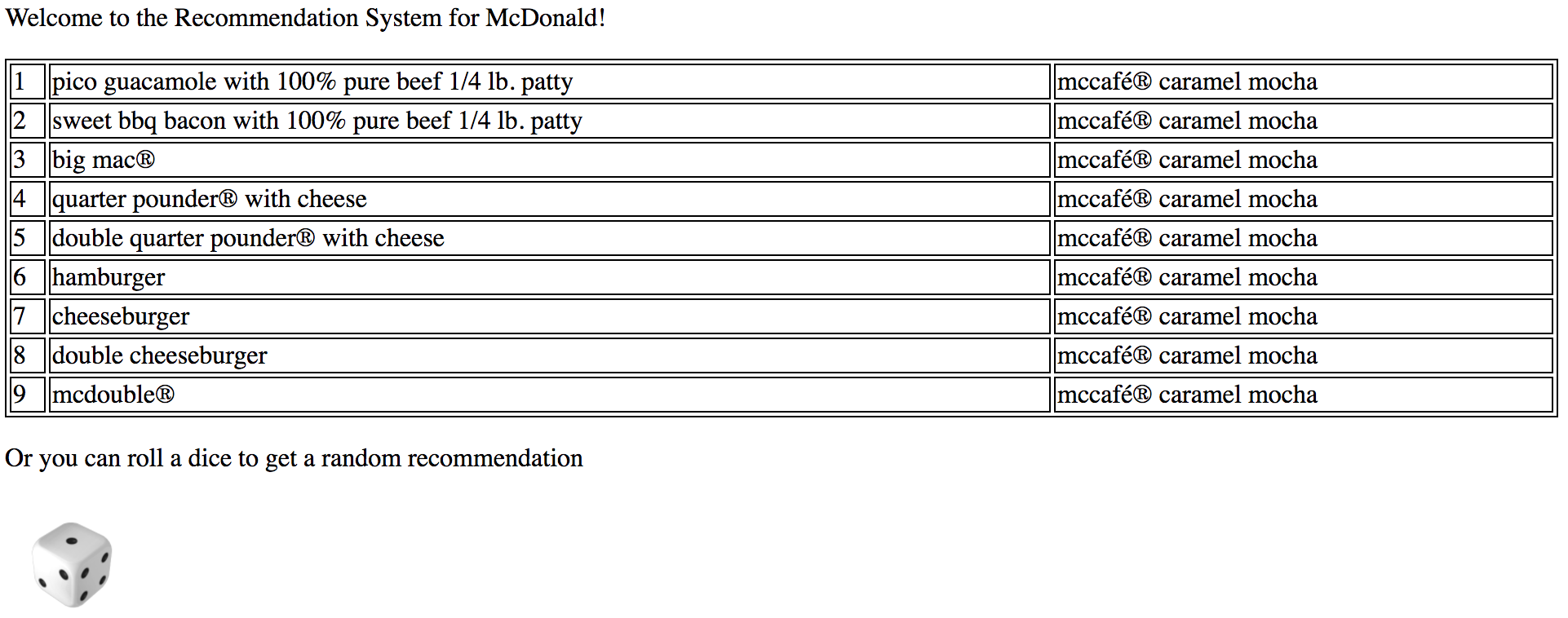
For better user experience, I designed an interface with drop down options to offer recommendations for users. The recommendations are based off of vegetarians, chicken, or beef options. Users can also decide which dessert or drink to have. Finally, they can select the nutrition type and the corresponding threshold. When clicking the “go” button, they will see a table of food recommendations based on their choices.

To implement it using Python code, I chose to use the web framework Flask for URL routing and web templates. The router.py maps three different routes. The first maps URL “/” to function “display()”. This function finds the template “display.html” in the templates/ folder, and pass different options as list on to the web template, renders it to HTML, and sends it back to the browser. Then, when users have selected the options and click ‘go’ button, the Python code use “request.form.get()” function to retrieve these options and pass the corresponding recommendations to the template “result.html”, which shows a table containing all recommendations based on the requirement. Finally, the user can click the Dice image (which directs to “get\_recommendation.html”) to get a random recommendation(using “random.randint()” function to get the index).

**display.html**



**result.html**



**random.html**

