

Bunch of Lunch Munches: Jeremy Kwok, Anthony Sun, Brianna Tieu, Vivian Teo
Soft dev
P04
2023-05-03

Target Ship Date: May 20th

Roles and Task Breakdown:

PM - Brianna

Effect PP Mode:

Front-end (HTML/CSS, Bootstrap, JS, Google Maps): Vivian, Brianna

Back-end (Database, Flask integration, parsing, Google Maps): Jeremy, Anthony

Dataset: [New York City Restaurant Inspection Results \(NYC Open Data\)](#)

Idea: Find out the health violations/inspection status of your favorite restaurants!

- Restaurant Search
- Display restaurant inspection results
- Can use Google Maps to pinpoint location of restaurants as well (using Google API).
- Users can save/favorite certain restaurants so they can easily see the information in another page

Components:

- Flask app
 - `__init__.py` - serves HTML pages with stored user information
- HTML
 - **login.html** (stretch goal) - landing page with login form and registration form
 - **home.html** - where users can search for a restaurant and view the list of results, along with a map presenting the locations of the results
 - search bar
 - results will be shown in the form of cards
 - each of the restaurants returned as a result will include the name, address, rating, a favorite/unfavorite button, and a button to see reviews.
 - the address for each of the restaurants will be displayed to avoid confusion when it comes to multiple restaurants with the same name.
 - clicking on one restaurant will replace the results with more specific information about that restaurant
 - “go back to results” button that’ll take you back to the results after viewing one restaurant
 - button that takes you to favorites page

- **saved.html** - users can view their saved restaurants and view each of the restaurant's inspection details
 - users can click the star next to the restaurant name to remove it from their saved restaurants.
 - button to go back to home page
- CSS
 - **style.css** - includes all the stylistic elements to the front-end using Bootstrap
- JavaScript
 - **script.js** - adds all necessary dynamic functionality to served Flask app
 - managing buttons and display results
 - stretch goal: icons that show the ratings (A,B,C,D,F) of restaurants on Google Maps
- API keys
 - **key_api** - each of the API keys
- Backend
 - SQLite population of database with restaurant information, including restaurant information and their inspection results
 - SQLite population of database with user information, including their saved restaurants
 - Python files dealing with API handling and database parsing

Database Organization:

- 1 table with user information
 - **users.db:** stretch goal
user id | username | password | list of saved restaurants
- 1 table with restaurant information
 - **restaurant.db:**
CAMI (unique restaurant ID) | DBA | borough | building | street | zip code | cuisine description | inspection date | violation code | score | grade

API:

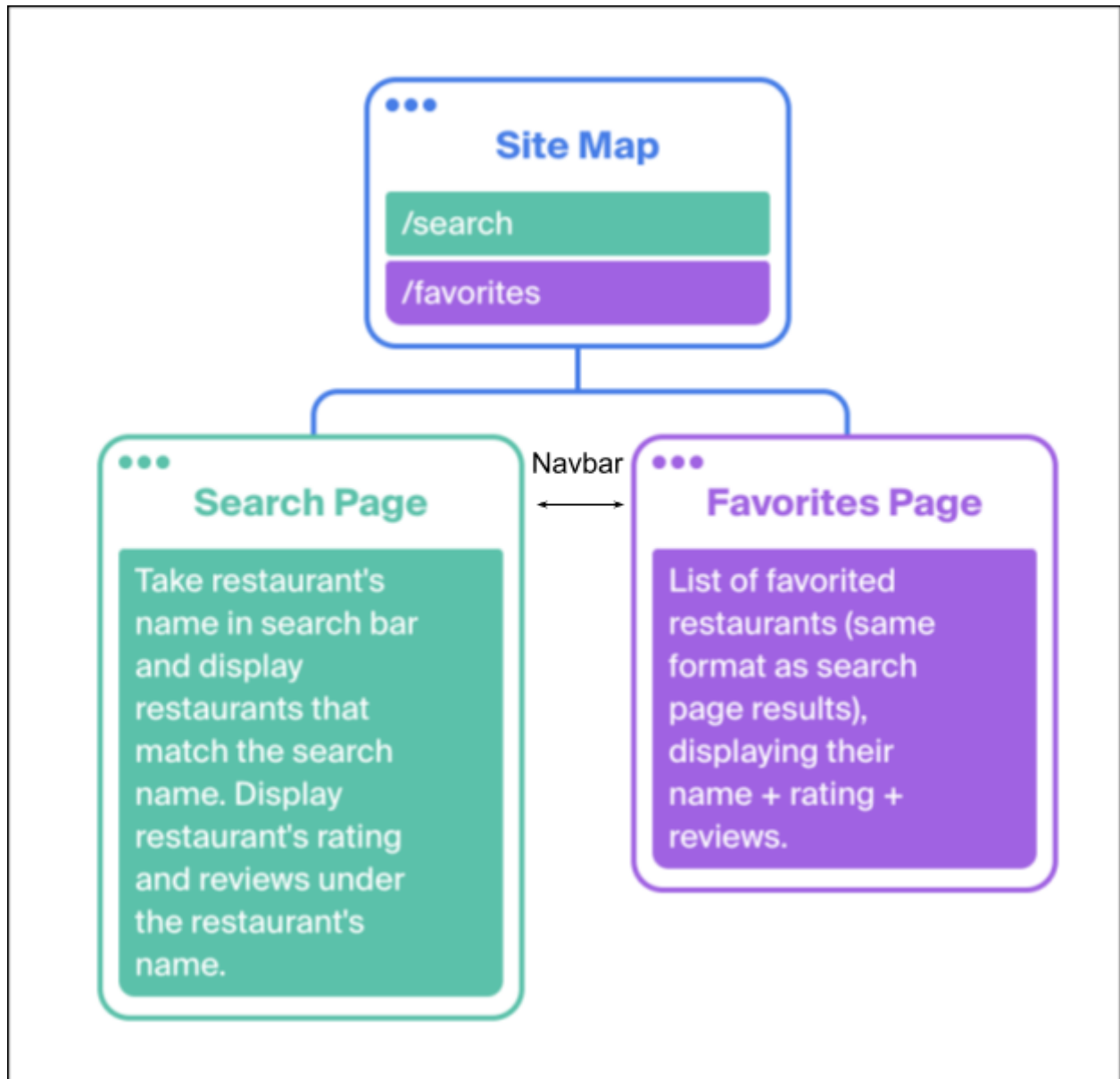
- Google Maps API:
 - Markers, Custom markers
 - Controls (center, zoom)
 - Simple click events (select)
 - Restaurant reviews

Front End Framework:

- Bootstrap:
 - preferred aesthetic

- has prebuilt elements such as cards (necessary for displaying our results)
- pre-styled forms for filtering and search bar

Site Map



Mockups:

Search Page:

CLEAN RESTAURANTS

SEARCH....

HERE'S WHAT WE FOUND:

★ POPEYES
• GRADE: • RATINGS (?)
★ MCDONALDS
• GRADE: • RATINGS (?)
★ UHH MORE NAMES
• GRADE: • RATINGS (?)
★ RESTAURANT NAME
• GRADE: • RATINGS (?)

CLEAN RESTAURANTS

SEARCH....

~~HERE'S WHAT WE FOUND:~~





★ POPEYES
• GRADE: • RATINGS (?)
★ MCDONALDS
• GRADE: • RATINGS (?)
★ UHH MORE NAMES
• GRADE: • RATINGS (?)
★ RESTAURANT NAME
• GRADE: • RATINGS (?)

clicking McDonald's will replace results w/ :

★ MCDONALD'S
GRADE:
Rating
Reviews:
...
other specific info

Favorites Page:

SAVED RESTAURANTS:

 <u>POPEYES</u> • GRADE: • RATINGS (?)
 <u>MCDONALDS</u> • GRADE: • RATINGS (?)
 <u>YHH MORE NAMES</u> • GRADE: • RATINGS (?)
 <u>RESTAURANT NAME</u> • GRADE: • RATINGS (?)

Priority List

High

1. Parse dataset and populate database with necessary info
2. Search for restaurant based off database
3. Google Map Integration

Medium

- Filtering restaurants (for example by ratings)

Low

- User login