Foodies Web Application

Introduction:

I love food and like to explore restaurants. I use google map to plan for my restaurant exploration. I always wanted to create an application that lets people share their dining experiences. The application can also let users make a reservation and get detailed instructions on how to get to the restaurant. That is how Foodies was created.

Distinctiveness:

Foodies is a web application that lets users share their favorite restaurants and their dining experiences. Users can add a new restaurant if it is not already in the application. If the restaurant is already in, they can write a review to share their dining experiences.

Each restaurant has its own information page. The page contains a leaflet map that plots the restaurant's location. The map is interactive. It supports zooming in and out. The user can also enter an address to get detailed driving instructions to get to the restaurant.

The page also contains reviews that users have shared about the restaurant. It also has a "Reservation" button. Users can use it to reserve a table. After the reservation is made, the user will be redirected to the restaurant's calendar. The calendar has all the reservations for the user. Each reservation is a link. Clicking on the link allows the user to edit the reservation.

Complexity:

Foodies was built using Django as the back-end and HTML, CSS, and JavaScript as the front-end. Other than the user model, there are 6 additional models. They are Profile, Category, Restaurant, Post, Follow, and Event.

Profile model lets users update their usernames, phone numbers, and profile pictures. Category model contains the description of food. Restaurant model accepts inputs from users such as name and address. Reviews go through the Post model and get displayed to users. Follow model keeps track of followers for each user. Event model handles the detail of a reservation.

Media URL and media root were added to settings.py to allow images to be stored in the application. Pillow was installed as an image library to handle images. Profile model has an image field. It allows users to upload an image and store it in the media root. Media URL turns the uploaded image into an URL before displaying it to users.

Geopy is used in views.py to turn a restaurant’s address into a set of latitude and longitude. The restaurant’s information page has a JavaScript that plots the restaurant’s latitude and longitude on to a map. Users can interact with the map by entering an address to get turn-by-turn driving instructions, distance, and estimated travel time to the restaurant.

Reservation is made via Event model and get populated as a clickable link on a calendar. Calendar is set up as a class in utils.py that inherits from HTML calendar. It gets called from CalendarView in views.py to create a calendar.

What is contained in each file:

Restaurant/urls.py sets routes to index, login, logout, account, create a review, restaurant profile, and calendar.

Views.py contains functions to process inputs from users or run queries to pull data from the sqlite3 database. The functions render html pages to display the processed information and show them to the user.

Forms.py set a class of event form to handle a reservation.

Utils.py set a class of calendar to take year and month as arguments. It returns html elements and passes them to calendar.html to display a calendar to the user.

Map.js has a function to take a restaurant’s latitude and longitude and plots them as a marker on a leaflet map. The base map is ArcGIS. The required api key is included in the file to show the map.

New\_restaurant.js has a function to prevent a form from being submitted by disabling the submit button initially. The submit button will be enabled if both name and address are filled in.

Post.js has a function to monitor the “edit” and the “like” buttons of a post. If the “edit” button is clicked, it changes the content of the post to a text box pre-filled with the original content. The user can make changes to the content and click the save button to save the changes. If the “like” button is clicked, it will increase the like count by 1 if the user didn’t like the post before. Otherwise, it will decrease the like count by 1.

Restaurant.js has two functions. The first function is to prevent a form from being submitted by disabling the submit button initially, like the function in New\_restaurant.js. The second function determines which restaurant option is selected. It passes the option as an argument to restaurantInfo in views.py to run a query to pull the restaurant’s address. The result of the query will be sent back to the restaurant\_info function as a Json response and append to a ‘div’ inside the html to display to the user.

How to run your application:

1. In the terminal, cd into the foodie directory.
2. Run python manage.py makemigrations restaurant to make migrations for the restaurant app.
3. Run python manage.py migrate to apply migrations to the database.