# Foodies Web Application

## Introduction

I love food and enjoy exploring restaurants. I often use Google Maps to plan my restaurant visits. The idea of creating an application where people can share their dining experiences has always intrigued me. Thus, Foodies was born.

## Distinctiveness

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

Each restaurant has its own information page, featuring a leaflet map that displays the restaurant's location. This interactive map supports zooming in and out. Users can also enter an address to get detailed driving instructions to reach the restaurant.

The restaurant's information page also includes user reviews and a "Reservation" button. Users can utilize this button to reserve a table. Once a reservation is made, users are redirected to the restaurant's calendar, where all their reservations are listed as clickable links. Clicking on a link allows users to edit the reservation.

## Complexity

Foodies was developed using Django for the backend and HTML, CSS, and JavaScript for the frontend. In addition to the user model, there are six additional models: Profile, Category, Restaurant, Post, Follow, and Event.

The Profile model allows users to update their usernames, phone numbers, and profile pictures. The Category model contains descriptions of food. The Restaurant model accepts user inputs such as name and address. Reviews are managed through the Post model. The Follow model keeps track of followers for each user, while the Event model handles reservation details.

Media URL and media root were added to settings.py to enable image storage within the application. Pillow was installed as an image library to handle images. Geopy is used in views.py to convert a restaurant's address into latitude and longitude coordinates.

Reservations are handled through the Event model and are displayed as clickable links on a calendar. The calendar is implemented as a class in utils.py, inherited from HTML calendar, and called from CalendarView in views.py to generate the calendar.

## File Structure and Functionality

- \*\*Restaurant/urls.py\*\*: Sets routes to index, login, logout, account, create a review, restaurant profile, and calendar.

- \*\*Views.py\*\*: Contains functions to process user inputs or run queries to retrieve data from the SQLite3 database. These functions render HTML pages to display processed information to the user.

- \*\*Forms.py\*\*: Defines a class of event form to handle reservations.

- \*\*Utils.py\*\*: Defines a class of calendar that takes year and month as arguments, returning HTML elements passed to calendar.html for displaying to the user.

- \*\*Map.js\*\*: Includes a function to plot a restaurant's latitude and longitude as a marker on a Leaflet map. The base map used is ArcGIS.

- \*\*New\_restaurant.js\*\*: Implements a function to prevent form submission by disabling the submit button initially. The button is enabled if both name and address fields are filled in.

- \*\*Post.js\*\*: Monitors the "edit" and "like" buttons of a post. If the "edit" button is clicked, it changes the content of the post to a pre-filled text box. Clicking the save button saves the changes. If the "like" button is clicked, it increases the like count by 1 if the user didn't like the post before; otherwise, it decreases the like count by 1.

- \*\*Restaurant.js\*\*: Includes two functions. The first function prevents form submission by disabling the submit button initially, similar to the function in New\_restaurant.js. The second function determines the selected restaurant option and passes it as an argument to restaurantInfo in views.py to run a query for the restaurant's address. The result is sent back as a JSON response to the restaurant\_info function and appended to a 'div' inside the HTML for display to the user.

## How to Run Your Application

1. In the terminal, navigate to the foodie directory.

2. Run `python manage.py makemigrations restaurant` to create migrations for the restaurant app.

3. Run `python manage.py migrate` to apply migrations to the database.