Restaurant Recommendation System

Team Members:

Heet Vora

Nishee Shelat

Project Overview:

Restaurant Recommendation System (RRS) is a web application which enables us to get the restaurants filtered based on location as city and cuisine and also the maps view of it through the latitude and longitude values. Consumers today gain a lot from apps that recommend restaurants. It is extremely convenient to be given a list of restaurants that fit our criteria without having to click through numerous reviews for each establishment or compare and contrast them. The web application majorly focuses on the state of Colorado and the cities in it. We acquired the database for restaurants in our application through Open Street Maps. When a user searches for a specific city in Colorado and for a specific cuisine in that city, they will be provided a filtered list of restaurants. Additionally, the user can look at its exact location in the maps as well.

Project Goals:

In this project, users can explore information about the restaurants on the system, including searching for restaurants using filters like city and cuisine. The system's goal is to help users find recommendations for fantastic local eateries. This system is made to look through the data you give and respond with a list of all the restaurants that fit the customer's criteria. This project's learning objective is to utilize at least four cloud technologies specified in the project criteria.

Software and Hardware Components:

1. **Frontend: HTML, CSS**

The process of developing a graphical user interface (GUI) for a website by utilizing markup languages such as HTML and style sheets such as CSS is referred to as front-end web development. Users are able to browse and engage with the website in this manner thanks to this feature. HTML is in charge of generating the document structure of a webpage, whereas CSS is in charge of adorning the webpage with various styles and formats.

2. **Backend: Flask**

Python is the language that was used to write the Flask micro web framework. It is referred regarded as a microframework due to the fact that it does not need any specific tools or libraries to function. It does not have a database abstraction layer, a form validation layer, or any other components that would normally be provided by pre-existing third-party libraries for the same duties. Flask, on the other hand, has support for extensions that let developers add functionality to their applications just as if the functionality had been built into Flask itself. There are extensions available for object-relational mappers, form validation, upload handling, a variety of open authentication protocols, and other utilities connected to standard frameworks.

3. Database: Firebase

Google Firebase is an application development platform that is powered by Google that gives developers the ability to construct applications for iOS, Android, and the Web. Firebase provides tools for designing marketing and product experiments, as well as for measuring data, reporting app faults, and fixing them. It also provides tools for fixing app errors. Flask makes ideas, but it does not impose any dependencies or layout requirements on the project. The developer is the one who decides which tools and libraries to employ; this choice is entirely up to them. There are many extensions that have been developed and offered by the community, which makes it simple to add additional features.

4. REST API's:

An application programming interface, sometimes known as an API, is a collection of definitions and protocols that is utilized in the process of developing and integrating application software. It is frequently referred to as a contract between a provider of information and a user of that information, specifying the content that the consumer (the call) and the producer (the producer) are supposed to supply to each other. When two different services need to communicate with one another, REST APIs are used for that purpose, while RPC is the protocol that is used to notify the service provider when a booking is made. We made use of it due to the fact that it is easy to operate and provides seamless connectivity with various other components.

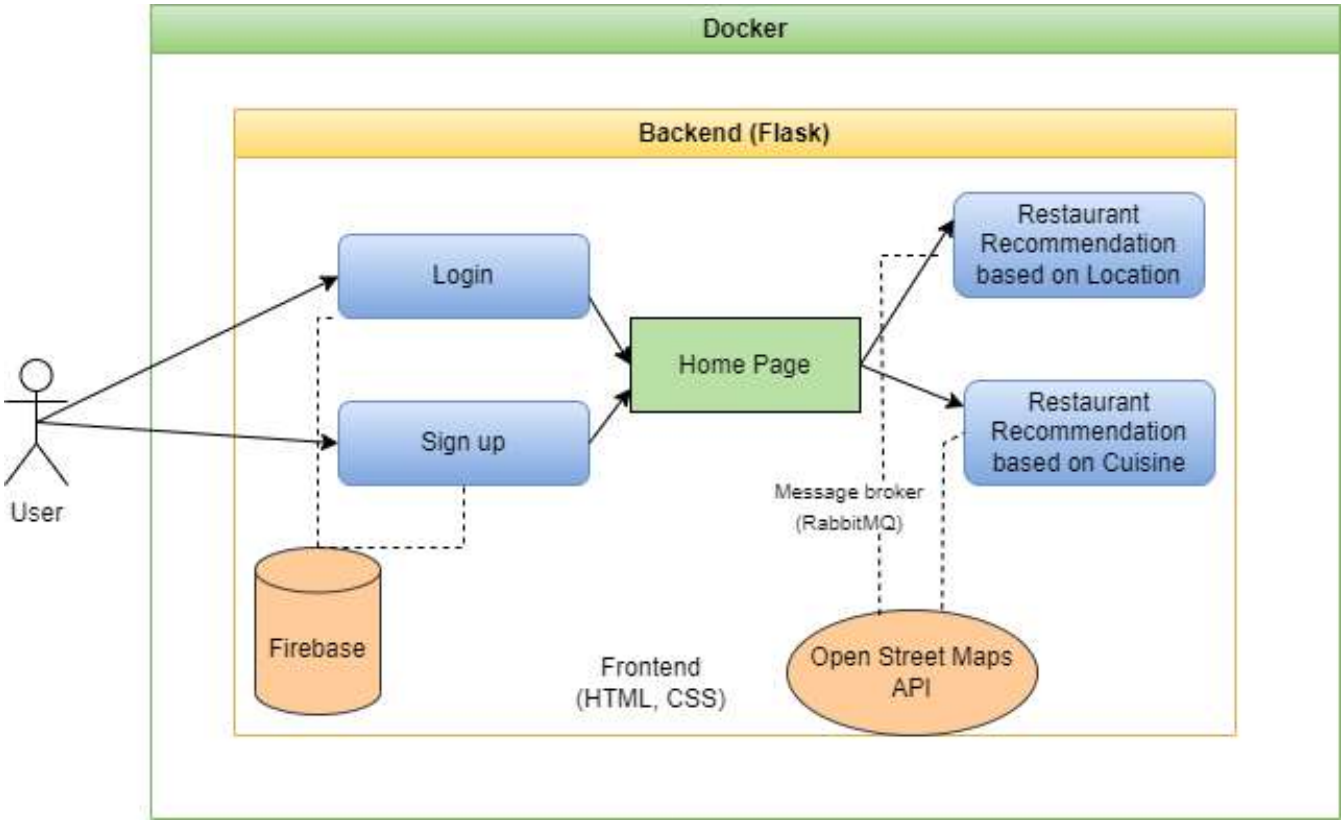
5. Message queuing: RabbitMQ

Message queuing is accomplished with the help of RabbitMQ. The term "queue manager" can also be used to refer to a "message broker." To put it another way, software is what creates queues and gives programs the ability to connect to them in order to send a single message or several messages. There is no limit to the variety of information that can be included in a message. Due to the stability of RabbitMQ's queues, the decision was made to utilize the platform. Troubleshooting RabbitMQ was the step that took the most time out of all of the processes.

6. Deploying: Docker

Docker is used across the entire development lifecycle with the purpose of developing applications that are quick, simple, and portable on both desktop computers and in the cloud. It eliminates monotonous and repetitive configuration procedures. Docker's all-encompassing end-to-end platform features graphical user interfaces (UIs), command line interfaces (CLIs), application programming interfaces (APIs), and security that are designed to cooperate with one another throughout the whole application delivery lifecycle. We containerize each freshly generated service with the help of Docker, which is the tool that we use. In the beginning, it makes deployment to cloud settings much easier to accomplish. With the help of these containers, we may generate environments that are reliable and distinct from those produced by other apps

Architectural Diagram:



Interaction between the components:

From the front end of the website, the user attempts to either log in as an already registered user or sign up as a new user. The application will only permit the user to log in if the user's email address and password already exists in firebase. If the user makes an attempt to sign in, a new email address and password will be stored in the Firebase database. Using the OpenStreetMap API, the application makes recommendations to the user on Restaurants based on Location as well as Cuisine. On the map, the locations of the recommended restaurants are also displayed, and this is done with the help of the message broker: RabbitMQ. When it is no longer necessary, the user can also log out of the application.

Debugging:

In the beginning, we begin by testing everything on our local PC and figuring out what the issue is by using logs and breakpoints. After we have containerized them, we will be able to employ a logging service to capture data in a text file concerning how they are being operated. Postman was the tool that we utilized in order to validate the operation of our APIs (Open Street Map API) and ensure that the data was successfully fetched. We utilized Jupyter Notebook to dissect our code and validate its individual components to determine whether or not they were operating as expected. As a result, we were able to avoid running our entire

program just to test certain features. We also used Visual Studio Code Debugger and stack overflow for any Docker related issues.

TESTING :

We will always have operational test scenarios, and we will examine all of the functionalities ourselves to ensure that they are operating as expected.

Capabilities of the Project:

- User can look at the exact location of the restaurant through the longitude and latitude values.
- Search for any restaurant in the city of Colorado.
- Look for restaurant on the basis of cuisine.
- User can login and logout after his/her session is done.

Limitations of the Project:

- Because of the size of dataset, we only restricted the search for Colorado state but had we used caching, we could have used the entire dataset of OSM.
- We can also add review feature in our recommendation system, through which the user can leave his/her review if they want to.

Future Scope of Project:

In the future, the scope of the project will include the creation of a greater variety of features, through which we will be able to improve the effectiveness of the process of making recommendations. We also have the option of requiring people to provide written feedback about a restaurant, in addition to providing ratings, and also posting photographs of the establishment's ambiance. To further enhance the quality of the recommendation system, we could also perform sentiment analysis on the input offered by the user.