FeastAI: An ML & LLM-Powered Dinner Selection Web Application

by

Rifat Mahamud Mukul
22141007

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
April 2026.

© 2026. BRAC University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I have acknowledged all main sources of help.

Student's Full Name & Signature:

Rifat Mahamud Mukul

22141007

Approval

The project titled "FeastAI: An ML & LLM-Powered Dinner Selection Web Application" submitted by

1. Rifat Mahamud Mukul (22141007)

of spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in spring, 2026 Year.

Examining Committee:

Supervisor:
(Member)

Saadat Rafid Ahmed

Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Arif Shakil

Senior Lecturer
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi

Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

In today's fast-paced Restaurant Industry of Bangladesh, discovering the perfect diner has become an experience that goes beyond the basic nature of searching; that is why a personalized and intelligent recommendation system has become essential to help Bangladeshi users to come up with an intuitive and effective way of searching restaurants. This project introduces a comprehensive restaurant recommendation engine that uses advanced machine learning algorithms and large language models (LLMs) to find the best eating options for each user. By assessing crucial characteristics such as geographical proximity, the system narrows down options based on the user's distance from probable restaurants, assuring convenience. After that, it uses sophisticated sentiment analysis and rate evaluation algorithms to analyze restaurant reviews, providing unbiased information on the caliber of the cuisine and the level of service. Furthermore, the engine incorporates user history and behavior tracking, learning from previous choices and dining patterns to recommend restaurants that match individual tastes. The unique time-based rating feature of this project ensures that suggestions are both enticing and useful by balancing the user's available eating time with the restaurant's food preparation time. In order to customize recommendations to each user's unique time limitations and culinary tastes, the system also looks into connections between food type and preparation time. The ultimate result is a strong, data-driven recommendation system that dynamically aligns high-quality dining experiences with both personal preferences and practical logistical issues.

Keywords: Restaurant Recommendation System, Personalized Recommendations, Topics covered include LLMs, machine learning algorithms, user preferences, sentiment analysis, real-time data, and dietary restrictions. Features include geolocation, dynamic recommendations, food preparation time, culinary preferences, user history, a data-driven system, and context-aware recommendations.

Acknowledgement

In the beginning, I'd want to thank Allah for His blessings, wisdom, and kindness that have given me constant strength during this path. This would not have been feasible without His instructions. I am grateful to my parents for their unwavering love, support, and encouragement throughout my academic and personal lives. I owe my success to their hard work, intelligence, and trust in me. I am grateful for their support. I am grateful to my supervisors for their guidance, patience, and valuable feedback that helped me complete my study successfully. Their expertise and guidance have been crucial during this process. I am thankful to BRAC University and those who provided resources, data, and assistance for this effort. Their partnership has made this vision a reality. Finally, I'd want to thank my friends and coworkers for their support and assistance in finishing this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

By helping customers locate meal alternatives that suit their likes, preferences, and dietary needs, the restaurant suggestion system project aims to satisfy a growing demand in the contemporary food industry. The difficulty is in effectively processing and comprehending the growing amount of user-generated content on restaurant review platforms and the accessibility of geolocation data. There are now more chances to enhance these recommendation systems because to the development of large language models and machine learning. By taking into account each user's interests and behaviors, the system aims to tailor recommendations. It also aims to be dynamic and contextually aware, responding instantly to factors like location, time, and restaurant availability.

This system makes use of geolocation to propose local restaurants and natural language processing (NLP) techniques to understand textual data from user reviews. The system is designed to be easy to use and provide real-time, individualized recommendations. The project performs much better than conventional recommendation engines by leveraging machine learning algorithms and LLMs, offering a much more accurate and rich user experience.

1.2 Rational of the Study or Motivation

Due to the increased dependence on online services for restaurant discovery, there is an increasing amount of structured and unstructured data that has to be effectively evaluated in order to produce trustworthy recommendations. Conventional methods usually concentrate on content-based filtering (restaurant-based recommendations) or manual filtering (user-based recommendations), but they are not very good at handling complicated queries or incorporating dynamic, real-time data. This project fills the gap by combining massive language models with sophisticated software approaches to enhance contextual relevance and suggestion customisation. This project aims to close the gap between current recommender systems and the need for a more sophisticated, user-focused, and flexible platform that can accommodate a variety of user requirements. Additionally, the demand for real-time restaurant information, including availability, hours of operation, and specials, as well as the growth of mobile applications, present a chance to create a system that is both

constantly updated and tailored. In such markets, which are rapidly growing each day, adaptability and relevance are key factors in that type of market.

1.3 Problem Statement

The purpose of the research is to develop a system that is customizable, context-aware, and user-friendly. Existing restaurant recommendation systems have a lot of issues, such as failing to figure out the customer's preferences, and location.Many existing systems just rely on old databases, expensive google map api or building the entire map by themself. All of these options are expensive, time consuming and make the service costly for the end users. So most of the existing platforms avoid this part but it creates some serious issues. Because sometimes the restaurant review posts are continuously bad or customers report the food issues. And also sometimes the existing platform doesn't even remove the low review restaurant or low rated restaurant. The platform just gives general instructions but that's not enough for the end users. Though this low review and low rated restaurant exist in the platform a big number of times users get this suggestion of those restaurants which makes them unhappy.

1.4 Objective

Designing a context-aware and highly personalized and customizable, and user-friendly system by implementing a RAG architecture and ranking algorithm algorithm is the main goal of this restaurant recommendation system. For recommending the personalized restaurant, the system uses the location, dietary and available time of the user.

1. To obtain the most recent recommendations, use real-time data, such as restaurant availability, promotions, and operating hours.

2. To increase the accuracy of suggestions, employ natural language processing (NLP) techniques to read and evaluate user reviews and restaurant descriptions.

3. Provide an intuitive and quick suggestion system with an easy-to-use UI.

By achieving such goals, the system will significantly enhance the user experience and support nearby companies by offering customized restaurant suggestions based on user preferences and present circumstances.

1.5 Scopes and Challenges

One excellent method for obtaining valuable location-based data is to scrape data from Google Maps using BeautifulSoup and Selenium. However, there are certain difficulties associated with it, just as with every strategy. I'll go over the many facets of this approach below, including the challenges that were faced and the potential it presents.

1.5.1 Scopes

Automation of Data Collection

The major purpose of this approach is to automate the process of extracting data from Google maps. Selenium mimics real-world user behaviors including as navigation, scrolling, and clicking, enabling for huge data extraction with minimal user engagement.

Scalable Data Extraction

Web automation instruments like BeautifulSoup and Selenium can easily scale the data extraction method. The ability to effortlessly scrape several pages is very useful when working with large datasets or doing lengthy investigations.

Comprehensive Data Collection

Selenium's flexibility to work with dynamic web components on Google Maps allows for the collection of comprehensive data, such as location names, addresses, ratings, reviews, business hours, and contact details. This permits the creation of an extensive and rich dataset for research.

Data Structuring for Analysis

After extracting the raw HTML data, BeautifulSoup is used to parse and extract specific data points in a structured manner (like CSV). Additional analytics like sentiment analysis, rating distribution, and trend identification may then be performed using this organized data.

1.5.2 Challenges

IP Lock

Google Maps can detect and stop artificial scraping by using IP lockout mechanisms. Access to the website may be temporarily or permanently blocked if an excessive number of requests originate from the same IP address in a short amount of time.

Solution: Using a random time difference (seconds) to avoid this IP detection. Which simulates the human surfing activity and lowers the risk of discovery.

Dynamic Class Names

Because Google Maps employs JavaScript rendering and dynamic class names, class names may vary with each request or each time the website loads. This makes it difficult to accurately extract data using static class names.

Solution: More adaptable XPath or CSS selectors that rely less on dynamic class names can be employed. Furthermore, Selenium's explicit waits for items to load guarantee that all necessary data is accessible prior to extraction.

Missing Classes or Elements

Occasionally, the HTML structure may not include the desired data because of incomplete page loads, missing classes, or website constraints.

Solution: Error-handling mechanisms such as try-except blocks guarantee that the scraping process continues even when certain parts are missing. Additionally, dynamic wait durations might allow the website to fully load the content.

Chapter 2

Literature Review

2.1 Preliminaries

The restaurant recommendation system uses a variety of fundamental techniques, including collaborative filtering and content-based filtering, to propose restaurants based on user preferences and item characteristics. Hybrid models, which incorporate various approaches, have been more effective at making correct recommendations. Retrieval-augmented generation (RAG) uses real-time data, such as restaurant availability and current promotions, to improve the relevancy of recommendations. Additionally, feedback from users and restaurant characteristics are scanned using large language models (LLMs) like GPT-4, which extract context and sentiment to assist customize recommendations. While combined, these technologies allow the system to provide flexible, customized, and modern eating alternatives that better fulfill the preferences of users.

2.2 Review of Existing Research

Since the thoroughness and accuracy of the data collected have a direct impact on the quality of suggestions, data collection is an essential step in any restaurant recommendation system. Conventional systems for suggesting products get much of their data from explicit input, including user reviews and ratings. Customer opinions and input are important, but they can additionally be influenced or biased. False testimonials are becoming increasingly common; they can be created by businesses to enhance their reputation or by customers for self-serving motives. Because of this manipulation, particularly lessens the veracity of evaluations, customers find it more difficult to trust the guidance they rely on when making decisions about what to buy[3]. This type of evaluation provides rapid insight into a person's preferences and level of comfort. In contrast, implicit feedback, such as browsing patterns, prior interactions, and the amount of time spent on restaurant listings, may provide helpful information for a higher knowledge of client needs[1]. As recommender systems grow, more intricate sources are connected, such as third-party APIs that provide dynamic, live restaurant information such as location, hours of operation, and offers.[2]. Gathering information is now including unstructured data, especially from content generated by users like social media postings and online re-

views, thanks to the growth of large language models (LLMs). Large amounts of textual data may be analyzed and processed by LLMs, such as the GPT-4, which can gather context-relevant data, restaurant traits, and user feelings needed to provide personalised suggestions for improvement [6] [4]. likewise by using geographical information, restaurant systems that suggest restaurants can adjust their recommendations based on the user's actual location, making them both helpful and specific. By merging real-time data, LLM-enhanced processing of data, and traditional data sources, restaurant recommender systems may produce suggestions that are highly accurate and individualized [5].

A key stage toward developing trustworthy and robust restaurant recommendation systems is data cleansing. Common flaws in raw data, particularly data gathered from user reviews and online platforms, include inaccuracies, missing numbers, discrepancies, and irrelevant information. Such defects could impair the system's capacity to provide precise and beneficial predictions. For example, reviews of restaurants may include misspellings, uneven language, or irrelevant content that detracts from the overall impression. Because of biases and noise in user-generated data, pretreatment and cleaning methods are crucial to ensuring data quality. For analyzing textual input structured and appropriate formation is a must. Tokenization, lemmatization, stemming, and stop-word reduction are the common methods for converting input textual data into a usable and appropriate form for natural language processing[1].

But just formatting the text isn't enough for in large language model scenario. For getting the best outcome from a large language model, the data quality must be high and relevant to the model. A large dataset contains lots of irrelevant data, such as spam, ads, and irrelevant content. So when working with this kind of dataset, it is essential to ensure that this kind of irrelevant data is removed before using it on the model. Ambiguity and contradictory statement is another big issues. Here, sentiment analysis can help to reduce it. After eliminating all the issues in the textual data, only then can the data be used for input to a large language model. And only then can the model perform the best and give the best and desirable output [4] [2].

Data cleaning is one of the most important aspects of this recommendation system. It is related to removing irrelevant and false data from the database. This data cleaning is very important, especially for the algorithms that rely on real-time data. For this type of algorithm's false or incorrect data case can cause a fall in the quality of the output. And this might be the reason for the overall fall in prediction quality. To solve the data quality issues and get the best and up to information from the restaurant, many external APIs like Google Places can help[5]. To get the best and appropriate suggestion, the model needs to update the data of the restaurants. Wherever a restaurant makes any change in its menu, the model should get that information as fast as possible. This recommendation system will perform the best and provide the best and suitable output only the the data quality, like the data cleaning and up-to-date data, is ensured[1].

The successful implementation and performance of modern restaurant recommendation systems, particularly those that use sophisticated methods like large language models (LLMs) and large-scale machine learning models, relies greatly on vector databases. Multidimensional data, including embeddings representations of

data points in a continuous vector space—is meant to be stored and managed in a vector database. The embedding of words and models that use transformers like the GPT-4 are two methods used in restaurant recommendation systems to represent data as vectors, such as restaurant labels, feedback from customers, and meal preferences. Machine learning algorithms can do highly efficient similarity searches because to these vectors, that capture the semantic relationships between objects. To illustrate, based on past interactions, the system could generate a vector representation of a user's preferences and compare it to a large database of restaurant vectors to suggest which ones are appropriate [1]. Vector databases' biggest advantage is their ability to do quick homology queries; which makes them ideal for real-time systems for recommendations. The vector database presents the most pertinent restaurants based on vector similarity when a user seeks the system. As an example, the algorithm will choose restaurant embedding that most closely match the user's vegan preference vector if the user prefers vegan food. Therefore, it possible to provide recommendations that are more contextually relevant and modified, for taking into consideration both user preferences and restaurant parameters [2] [6]. In addition, multiple forms of data may be stored within vector databases, which enables the system to incorporate many data types (such as text, images, and user ratings) into a single, cohesive vector presentation. These strengthens the system's ability to adapt as well as capacity to manage complex inquiries that consider a variety of factors, such as user emotion, menu variation, and restaurant atmosphere.

Using methods that utilize LLMs, vector databases accelerate the extraction of pertinent restaurant data. Assisting LLMs to better comprehend the relationships among various restaurant attributes and client preferences, text-based data, such restaurant reviews or descriptions, may be converted into vectorized embedding. As the algorithm can take advantage of the vectors in the database to match people with restaurants that have conceptual similarities, this increases the degree of precision of the suggestions. Also, the system's ability to provide highly accurate and personalized recommendations becomes better as the database grows and more user interactions become recorded. As an outcome, vector databases boost the system's ability to grow and capacity to provide accurate real-time suggestions as data complexity and quantity grows [5].

By integrating the beneficial features of machine learning and retrieval-based techniques, the Retrieval-Augmented Generation (RAG) workflow marks an important step forward in recommendation systems. Standard recommendation systems often employ content-based or shared filtering. Whereas content-based filtering matches item qualities to user interests and shared filtering uses user interaction data to propose products based on like users, both approaches usually find it difficult to adjust to sophisticated and dynamic queries that need context and real-time information. For obtaining appropriate information from an external database or real-time APIs, RAG overcomes these limitations and uses it to provide customized recommendations [1] [4]. The RAG pathway in restaurant recommendation systems first chooses a group of possible eateries from a database according to the cuisine type, geographical areas, restrictions on diets, and client preferences. Following the finding of the appropriate restaurants, the findings are refined using a generative model, such as a large language model (LLM), which subsequently offers a final recommendation that analyzes other user-specific factors, such sentiment in reviews or previous preferences. The system is capable of offering highly customized and

contextually suitable recommendations because to this two-step technique, which combines retrieval and creation. It also takes note of both static preferences and dynamic, real-time factors like restaurant availability and offers [2]. Sophisticated user queries are a feature of the RAG workflow. When a user asks for a vegan restaurant within a given radius with a particular ambiance, for instance, the RAG pipeline first obtains a list of nearby vegan eateries before using the generative model to further customize the suggestions based on user reviews, restaurant ambiance, and other factors. RAG methods are exceptionally adaptable and equipped to provide real-time, contextually relevant suggestions because of their ability to combine retrieval and creation. Nevertheless, it keeps being a challenge to improve the RAG workflow to manage massive datasets as well as offer quick answers while still maintaining suggestion reliability. The approach will function even more effectively if the retrieval phase is adjusted to prioritize the most pertinent findings and the generative model's capacity of incorporating multi-modal input (such as photos, reviews, and location) is reinforced [5] [1].

An interesting method to improve the usefulness and individualization of restaurant suggestions is the Retrieval-Augmented Generation (RAG) pathway. RAG creates highly personalized and contextually relevant recommendations by combining the prediction ability of large language models (LLMs) with the retrieval of important restaurant data from various sources. The classical system of recommendations usually uses predetermined algorithms that might not be able to adapt to changing consumer needs or appropriately handle complex inquiries. RAG helps to integrate the real-time data with a generative model, which helps to generate the best and suitable, and modified suggestions for individuals. [2]. For improving performance, RAG offers multiple approaches that have several benefits. This allows refining the external data sources and optimizing and customizing the model. When the system collects the relevant data of the restaurant, it is the most important phase of the development. The majority of chances of failure or success of the model depend on this phase. For the ultimate best-performing model that provides the best suggestion can be ensured in the initial stage by prioritizing the collection of high-quality and up-to-date data. Additionally, the performance and the quality of the recommendation will be more effective when the data comes from multiple sources. Furthermore, whenever the model gains the ability to take multi-model inputs such as text, image together the performance will increase. By analyzing multiple data types together, the model can provide more relevant recommendations for each individual [4] [5]. The RAG pipeline is very crucial because it helps to improve computational efficiency. As it is a continuously growing system so it is also important for the model that it can handle this and provide the best outcome quickly. On the other hand, the designed model is also able to provide the best outcome with less amount of data. It helps the model to respond and handle more rare and unpredictable situations. By this process model also get the chance to familiarize yourself with the rare and unknown situations, which helps the model in the future. Moreover, continuously incorporating the rare and up to date and user feedback will help the model perform better[2] [5]. Restaurant recommendation systems can enhance consumer satisfaction and engagement by optimizing the retrieval and delivering parts of the RAG pathway to provide greater accuracy, dynamic, individualized suggestions [2]. The project makes itself exposed because, unlike other competing platforms that rely largely on premium services, it primarily uses free and dependable technologies and

APIs. This significantly minimizes operational expenditures, enabling the project to provide high-quality services at a lesser cost. As a result, end customers gain from higher quality services at a lower cost, providing the project a competitive advantage in the market.

2.3 Summary of Key Findings

Research already conducted indicates that composite systems of recommendation, which integrate content-based and collaborative filtering, offer more varied and reliable restaurant choices. By evaluating user evaluations and restaurant descriptions, large language models (LLMs) enhance customization by enabling the system to better understand user attitudes and preferences. A further development that is becoming becoming increasingly prevalent is real-time data integration, which enables the system to provide contextually appropriate recommendations based on the availability or promotions of the restaurant. Multiple findings imply that in order to give more accurate and customized suggestions and boost user satisfaction, future systems should keep combining different strategies and incorporating dynamic data.

Chapter 3

Requirements, Impacts and Constraints

3.1 Final Specifications and Requirements

Requirement analysis identifies required factors, demands from users, and features of the system for effective planning and execution. The following part highlights user and system requirements to ensure project success and develop a user-friendly recommendation platform.

3.1.1 User Requirements

- **Reliable Data Sources :** Data is the heart of the platform so users want the data to be from a reliable source and unbiased platform. For that google map is being used as a data collection platform for versatility and reliability. Because users can post any good or bad review to a restaurant on this platform and neither restaurant nor even google map hide it or remove it.
- **Personalized Analysis :** Every user needs personalized recommendation for each of their query for this the system uses text and selection both options for every user user just select or type their personal preference any then the system generates the best and the most suitable option for them.
- **Accessibility and Ease of Use :** The system has an interactive and easy to use visualization. So that any user can use it without any trouble. Users can select or type their preference and even users can use chatbot to get their preference.

3.1.2 System Requirements

- **Web Application Framework :** This is a python based web application so the platform uses flask. Which is best and most suitable for python.
- **Data Processing Pipeline :** From data collection to make the platform usable for the platform there was a pipeline. Which is made based on python beautifulSoup.

- **Database :** Raw data which is collected from google map and cleaned using selenium and python that data is stored on the database so MySql database is being used and then from the comparison the data convert into vector against each data point so for that storing vector database is being used.
- **Browser Compatibility :** All the modern browsers which support ES6 and CSS grid and flex are able to successfully run the web application.
- **Cross-Platform Compatibility :** The web based nature of the application gives it the freedom to run in Mac,Linux,Windows all the platforms.

3.1.3 Platforms and Tools Utilized

- **Flask :** For the backend and the api gateway flask is being used for its lightweight and easy to implement behaviour. It acts as an intermediary between the end users and the system. Its modular nature makes it for suitable options for the implementation of different ML algorithms
- **Python :** It is used as a primary programing language for the system. And different python libraries like numpy, pandas, matplotlip, and sklearn use the data to transform the raw data into the usable data for the system.
- **Ngrok :** This is used for secure tunneling and making it public for collecting the used feedback.

3.1.4 Computing Resource Requirements

- **Hardware Requirements :** The execution of the project requires the balance of hardware and software. Which ensures the smooth operation of real time recommendations. For the standard environment 64 bit multi core processor paired with minimum of 8 GB RAM is the best choice. For the storage requirements it requires a minimum of 5 GB fast ssd for the code dependency.
- **Software Requirements :** The core system is backed by Python (versions 3.9+), which fill all the necessary and essential libraries for data manipulation and machine learning. Dependency management is strictly handled through virtual environments to ensure version consistency. Ngrok is being used in the system to facilitate secure external access.
- **Performance Considerations :** The performance of the feastAI web application is primarily measured by retrieval latency and data throughput. Because of using the free services like LLM API the responsiveness of the system sometimes might be slowed down. Or even that it requires the regular API checkup to maintain the free API usability.

3.2 Societal Impact

The technology will enhance the dining experience by providing personalized restaurant recommendations based on the user's tastes, health needs, and location. It will encourage people to make healthier eating choices by recommending eateries that

cater to certain dietary requirements (e.g., vegan, gluten-free). From a sociological standpoint, it will boost **local businesses** by promoting a varied selection of eateries, including tiny and lesser-known ones, and will assist consumers in discovering new dining options. The system will also encourage diversity by making recommendations for various cultural dietary practices. Ethically, the system will respect user privacy by adhering to data protection standards such as GDPR and giving consumers control over their data. The implementation of open and trustworthy technologies and APIs substantially reduces operational costs while making the project more affordable to end users and adding to its social value by providing relatively cheap and high-quality services.

3.3 Environmental Impact

The manner in which the system keeps data and uses cloud services will influence how they impact the environment. Minimizing negative environmental impacts may be achieved by hosting on **carbon-neutral** and **sustainable** cloud platforms like AWS or Google Cloud. Additionally, minimizing the waste of water and food, and also collecting daily necessities from the local sources, are steps to adopt and promote a sustainable, environmentally friendly system. But long-term regular monitoring and maintenance are needed because the volume of the data is increasing day by day at an exponential rate.

3.4 Ethical Issues

Safeguarding personal information is the moral ground. There are some standard data protection laws to protect the data of the user. Such laws are the **CCPA** and **GDPR**. By implementing this law, the user will get access to their data even though the data is stored on the company's database. According to this law, user can delete and control access to their data. Additionally, the relevance and transparency of the suggestion of model are very important in the issues. In addition, it is also essential to ensure that the system will not encounterrage the user for any kind of food that is not healthy and safe for the human body. Utilizing free technology, the initiative bypasses the ethical dilemma of charging unreasonable costs to customers, making advanced services more accessible to a wider spectrum of individuals. This is also in alignment with a moral responsibility to give equal opportunity for all.

3.5 Standards

For the betterment of the user, the program will adhere to the Web Content Accessibility Guidelines as well as the data protection law. Implications of this law and accessibility rights make the platform secure and trusted by the user. Furthermore, the development of AI is growing day by day, so the use of ethical AI will enhance the performance of the platform and the trust of the user.

3.6 Project Management Plan

3.6.1 Data Preparation

Data Collection

I have collected all the data from the Google map, and to collect all the data i use Selenium and beautifulsoup. After collecting all the data i cleaned the data and made it ready for using into the project.

Data Preprocessing

Using different python libraries collected data is being processed for the final use in the machine learning algorithm and LLM.

3.6.2 Model Design

Framework Selection

For building the project i selected the Flask framework. Because it's the most effective and lightweight framework, which is also most suitable for the Python language. Another key reason for selecting flusk is that I developed the static RAG in Python.

Model Development

Using the selected framework i build a static RAG pipeline by inplimating haversing and cosign similarity formula.

3.6.3 Development Mode

Prototype Development

For monitoring the visual update i have created a prototype from the beginning of the project. So that I can track the update and also figure out the frontend and backend api issue as fast as possible.

User Interface (UI) and Experience (UX)

I have designed a UI/UX that prioritizes platform-specific usability and user-friendliness for the project. When adding a feature, I have considered the user preference, which will allow users accessibility over the platform.

Testing

To find out the efficiency of the prototype figeroer, our performance issues and bugs are important. To measure the real performance of the recommendation system after all the things test the prototype with real-world data.

3.6.4 API Integration

For the smooth connectivity of the frontend and the backend i have built and optimized the api so that within minimum of time, the query and response exchange between frontend and backend.

3.6.5 Design and Implement Database

Database Architecture

I have created a database to hold user interactions, model outputs, and processed data. Depending on the needs of the data structure, I heve select MySQL for the database.

Data Security and Access Control

For sensitive data for example user phone number, email and password i have used encryption before storing them into the database so that even it any issue arise the data remain protected.

3.6.6 Future Work

The future update of this web application ensures a more user centric approach and builds mobile applications.

3.7 Risk Management

Incomplete, inconsistent, or irrelevant data may be handled during the Data Cleaning step. To mitigate this, it is crucial to identify and handle missing data, remove duplicates, and detect outliers early. Misclassification during sentiment analysis or tokenization difficulties is a major risk during data preparation. To lessen this risk, meticulous dataset preparation and evaluation of sentiment categorization accuracy are required. It is possible the fact that a model that doesn't seem appropriate for the project will be chosen during the Model Selection phase. By evaluating many models and using parameters to guide the selection process, this might be mitigated. For Model Development, the risk of either overfitting or underfitting is substantial. By using appropriate methods of regularization and closely monitoring validation errors, it may be reduced. In Create RAG Pipeline, retrieval and creating components may not integrate properly. Through evaluating the process at each level and ensuring that components combine properly, this risk may be decreased. There is a possibility that data will be incorrectly segregated, which could result in biased or incorrect recommendations. Precise segmentation based on user behavior and attitude could mitigate this risk. Regarding Embedding, the primary concern exists that poor embeddings might impact the efficiency of the model. Using pre-trained embeddings and adapting them to the particular project dataset will improve the quality of the embeddings. The potential for misinterpreting insights in Insight Generation might lead to inaccurate advice. This issue will be resolved in part by continuous examination and verification of observations against user questions. Furthermore, the real-time recommendations in Deliver Insights and suggestions might

be delayed or wrong. This risk can be reduced by refining the recommendation algorithm and putting pre-computation or caching techniques into place.

Negative Impact	**Mitigation**
System crashes if any free service is stopped	Switch to a premium version or look for other sources.
AI model fails if the free API is terminated	Prepare for backup APIs or local processing capabilities.
A loss of data or service outage from the free MySQL server	Backups on a regular basis, as well as migrating to a more dependable provider

Table 3.1: Potential Negative Impacts and Mitigations

Since I use freely available technologies, there is a probability that anytime an organization stops offering free services, such as an AI API or MySQL server, the system may collapse, resulting in output stoppage or data loss. To avoid this, consider moving to premium services, maintaining frequent backups, and getting ready for substitute possibilities.

3.8 Economic Analysis

For the initial phase, the project has built on fully free tools. So that the user can access the best platform at no cost. In the current phase, the project uses free tools like beautifulsoup, selenium, python and most important free LLM api. But in fututе when the number of users gerows i will implement some paid options like a paid api for LLM to handle the big number of users. Also, in the upcoming I will introduce a very small number of subscriptions and alows restaurant to advertising there new food and services on the platform for which they also pay a little amount of money. This helps both the user and the restaurant pay less money for the services. Strategic planning and testing can help to reduce the project's inherent risks. The project may turn a profit within a reasonable timeframe with effective user acquisition and marketing, and its scalability guarantees room for expansion.

Chapter 4

Methodology

4.1 Methodology Overview

The method for web scraping, data cleaning, analysis, and visualization is described in the figure. First, pertinent data is scraped using Selenium/BeautifulSoup, then unnecessary information is eliminated, the data is formatted for analysis, and text analysis is applied. Using Python libraries such as Pandas and Matplotlib, the data is then stored in CSV files and then used for Exploratory Data Analysis (EDA). Finally, the enhanced data is stored in CSV format and graphs and summaries are generated. Then, the clean and usable data is used by the system to generate the best recommendation. Here, the system frist calculate the distance by the haversine formula and then uses cosine similarity for finding a list of the best matches. Then the list of best match restaurants and the user query are sent to the LLM to generate the rest response that matches the user's vibe.

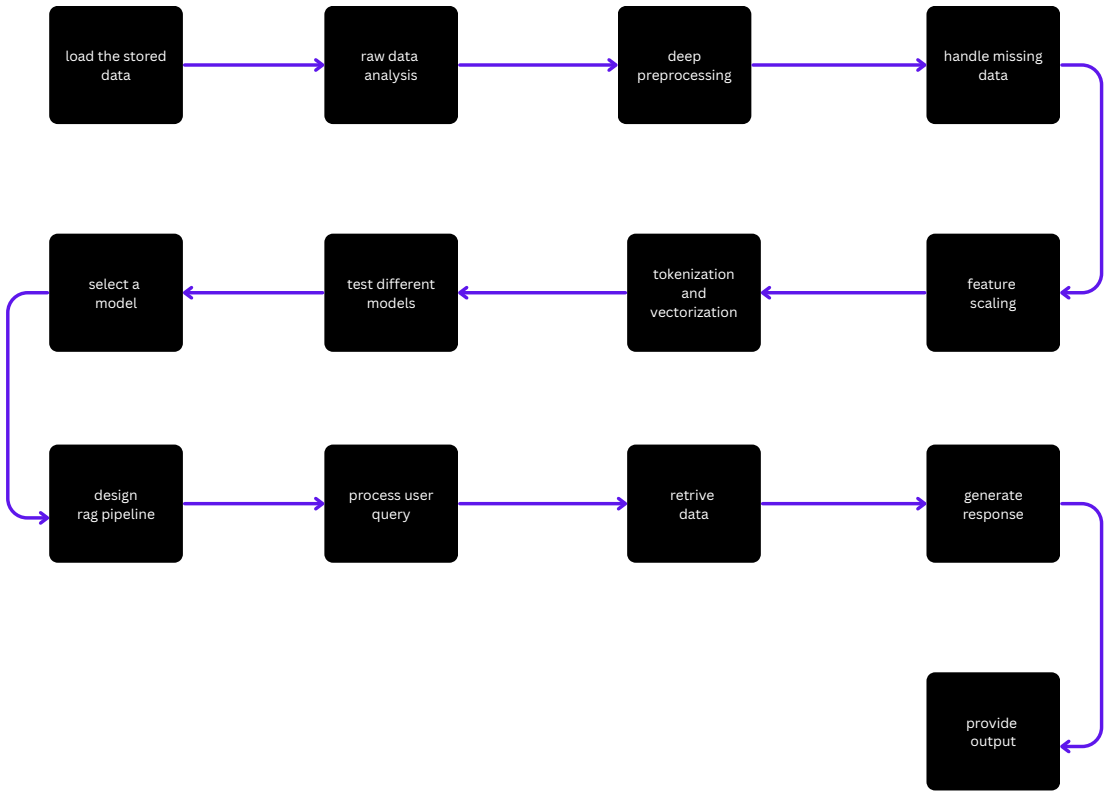

Figure 4.1: RAG Pipeline and Query Output

In the figure 4.1, data collection, analysis, and preprocessing are the first steps in a machine learning pipeline. It scales features, handles missing data, and gets the data ready for the model. After **training** and **evaluation**, forecasts are developed.

Retrieval-Augmented Generation (RAG) is used in this stage to enhance the technique by combining relevant data acquired from external sources with the model's knowledge. By generating customized responses based on **user preferences**, this improves the model's **relevance** and **accuracy**.

4.2 Preliminary Design or Design (Model) Specification

The static RAG is primarily used for personalized recommendation and based on used preference. The chain of operations will function as follows:

Restaurant Attributes

Each restaurant have a distinct set of attributes such as location, type of cuisine, ratings and average cost.

User Information

When the user query for the recommendation the system generates the recommendation based on the user preference and location coordinate.

Static RAG Integration

The static RAG pipeline play a major role in this system by

- **Retrieving relevant data**: Depending on the user's current location coordinate and restaurant coordinate, all the suitable restaurants are extracted from the database.
- **Augmenting with other sources**:Static RAG will take some additional info like time to wait or dine type for more accurate and personalized recommendation.
- **Generating tailored suggestions**: Using the information that has been provided static RAG presents one or more suitable options to the user.

Pipeline Design

User Inputs

- User provided his current location.
- Preferable cuisine type.
- Maximum amount of time the user have to reach the restaurant.

Restaurant Data

- Every restaurant's physical location, number of reviews, total rating, user review, average cost , contact and food type, are included in the database.
- This information is already pre-processed and arranged for convenience of retrieval.

RAG Process

- **Retrieval**: Using the user's location coordinates, the system retrieves a list of nearby restaurants.
- **Augmentation**: The system adds external sources to the restaurant data, such customer preferences (like past restaurant choices or favored cuisines) or real-time data (like the weather or special offers).
- **Generation**: RAG then combines user selections and the acquired restaurant data to offer personalized restaurant recommendations.

Personalized Output

The user is presented with a list of recommended restaurants that suit their preferences and are close to their current location.

Core Components

Location-Based Filtering

The user's current geographical spot will be used to shortlist restaurants.

RAG Model Integration

In order to make certain that the material is tailored to the individual's particular situation, RAG will be utilized to collect and provide recommendations.

Recommendation Output

A collection of restaurants will appear in a hierarchy of relevance, utilizing consideration factors such as proximity, ratings, cuisine type, and past user selections.

Technologies and Tools

Backend

- **Database**: The details of the restaurant is save.
- **API Integration**: Integrate location APIs to retrieve user coordinates and nearby restaurants.

RAG Model

- **Retrieval**: To obtain the details of the nearest restaurant, use the geographic coordinates.
- **Augmentation**: Utilize user choices, reviews, and historical data to improve the suggestions.
- **Generation**: Create customized restaurant recommendations using a trained RAG model.

4.3 Data Analysis

This section explains how the data is being collected from google map using selenium.

4.3.1 Data Collection

The initial data was extracted using Selenium, which automated the extraction of metadata from Google Maps depending on specified geographic regions. The info gathered consists of business details, user ratings, reviews, and operational hours, which are going to be utilized in the projects.

Figure 4.2: Restaurant info in html format

4.3.2 Data Cleaning

This phage deals with a massive number of unclear data that comes after scraping. In this phase the data is primarily cleaned by using BeautifulSoup and then for better understanding using python libraries.

4.3.3 Data Transformation

Methods for normalizing, scaling, or encoding data in preparation for analysis. Techniques may include:

- Normalization.
- Scaling.
- Encoding.

4.3.4 Data Integration

combining information from several sources as appropriate. This might entail combining datasets—whether relational or non-relational—from different internal or external sources.

4.3.5 Data Reduction

Methods for choosing pertinent characteristics or lowering dimensionality. Typical techniques may be:

- Principal Component Analysis (PCA) for dimensionality reduction.
- Feature Selection using techniques like importance ranking from models.
- To produce more significant features, use the technique of feature engineering.

4.3.6 Summary of Preprocessed Data

Detailed overview of the finalized dataset used for study and design. This includes its structure (including the number of rows, columns, and characteristics), any modifications performed, and its appropriateness for modeling or additional analysis.

4.4 Implementation of Selected Design

Implementing the selected design is the process of turning the theoretical idea into a functional system. This technique includes the following steps:

4.4.1 System Setup

The first step is setting up the environment, which includes selecting and configuring the required software and hardware platforms. This may include setting up databases or cloud services, as well as installing necessary frameworks and libraries.

4.4.2 Data Collection and Preprocessing

Gathering the required data comes next after the environment has been set up. Web scraping, APIs, and external databases are some of the possible sources of this data. The information is then cleaned up applying the techniques decided upon throughout the designing stage, including normalizing, correcting missing values, and scaling features.

4.4.3 Model Training and Optimization

Following all the information has been prepared, the model must be created and trained. To do this, a suitable machine learning or deep learning method must be selected, trained on preprocessed data, and then the model must be optimized by changing hyperparameters to improve accuracy.

4.4.4 Integration of Static RAG Model

This stage involves implementing the static RAG into the system. And this implantation will increase the result quality and also increase the user engagement. This Static RAG helps the user to get the best and most accurate and personalized suggestions.

4.4.5 System Testing and Evaluation

After the system is built, its accuracy, reliability, and performance are measured by standard testing baseline. This involves assessing the user interface, retrieval, and recommendation processes, as well as the system's capacity to handle various user input types. For achieving the best performance the evaluation is carefully handled and if there are any important and relevant evaluation comes then implement it.

4.4.6 Deployment

After validation, the system will be put in a real-world environment. It may include connecting the system to a web page or mobile application so users can interact with it in real time. The deployment procedure also includes keeping an eye on system performance and fixing any issues that may arise.

4.4.7 Maintenance and Updates

Frequent maintenance and upgrades are required to keep the system operating correctly after deployment. This could involve incorporating new features, updating the model with new data, as well as solving issues caused by feedback from users or technological glitches.

4.5 System Evaluation and Future Considerations

4.5.1 Current Status and Progress

- **Data Collection:** For collecting data from the google map selenium is being used, and also make sure that only the appropriate portion is scrap form the large block of data.
- **Data Cleaning:** For cleaning the raw data and making it impactful for the next phase beautifulSoup is being used. It is also to make sure that the extra html and irrelevant tag must be removed.
- **Exploratory Data Analysis (EDA):** A primary analysis is introduced for finding the problematic data that can be used for the project.
- **Database Schema Design:** A standard and suitable schema is designed and being used for storing the data.
- **Data Storage:** The clean and appropriate data being stored in the database for use in the project.
- **Query Optimization:** Optimized SQL query for efficient and less time consuming retrieval.
- **AI Model Experimentation:** Some AI models are tested by their api before implementing into the project.
- **AI Model:** For performance and robustness the best AI model is being selected.

- **RAG Pipeline Design:** A Retrieval-Augmented Generation pipeline was designed to increase the response accuracy by incorporating collected information with a selected AI model.
- **Error Resolution and Final Tuning:** In this project, for maintaining the quality and the performance of the system, important testing and error handling are being done. This includes :
  - **Code Review and Testing:** The currentness and fully working functionality ensures my checking every part of the code.

```
(appenv) rifat@rifathp:~/Documents/feastAI$ python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 647-047-248
127.0.0.1 - - [11/Feb/2026 21:51:28] "GET / HTTP/1.1" 200 -
```

Figure 4.3: Starting log of the system

  - **User Feedback via NGROK:** I use ngrok to allow users to interact with the system in real time and give me feedback.

```
ngrok                                                                    (Ctrl+C

Session Status                online
Account                       rifatMukul (Plan: Free)
Update                        update available (version 3.36.0, Ctrl-U to update)
Version                       3.35.0
Region                        India (in)
Latency                       57ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://e9ad-103-110-114-76.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Figure 4.4: Starting ngrok log

4.5.2 System Design

The system design describes the architecture, component, and the data flow of the application. The system allows the user to find the most suitable restaurant based on their location, vibe and availability of their time. This application scraps a huge amount of data from the google map and processes it and then uses AI to recommend the best restaurant to the user.

- **Use Case Diagram :** This Use Case Diagram 4.5 illustrates the interactions between the "User" and the system. Here, users can perform several actions, beginning with registering themselves on the system, logging into the system, and asking the system for recommendations. The system also provides the user with the best possible outcome by applying some filtration and some advanced techniques.

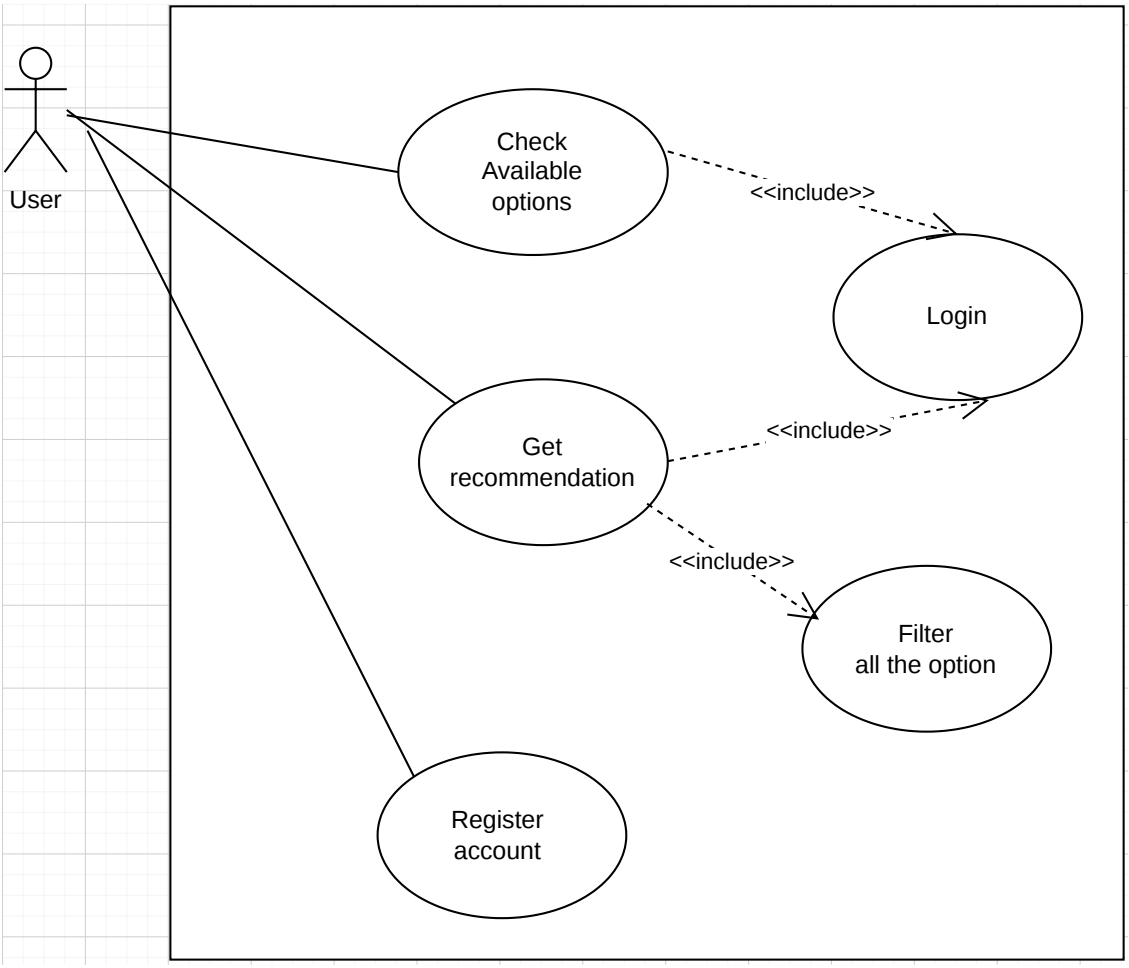

Figure 4.5: Use Case Diagram

- **Sequence Diagram :** The diagram 4.6 shows how the user request and the recommendation works sequentially. It includes frontend, backend, database and the LLM.

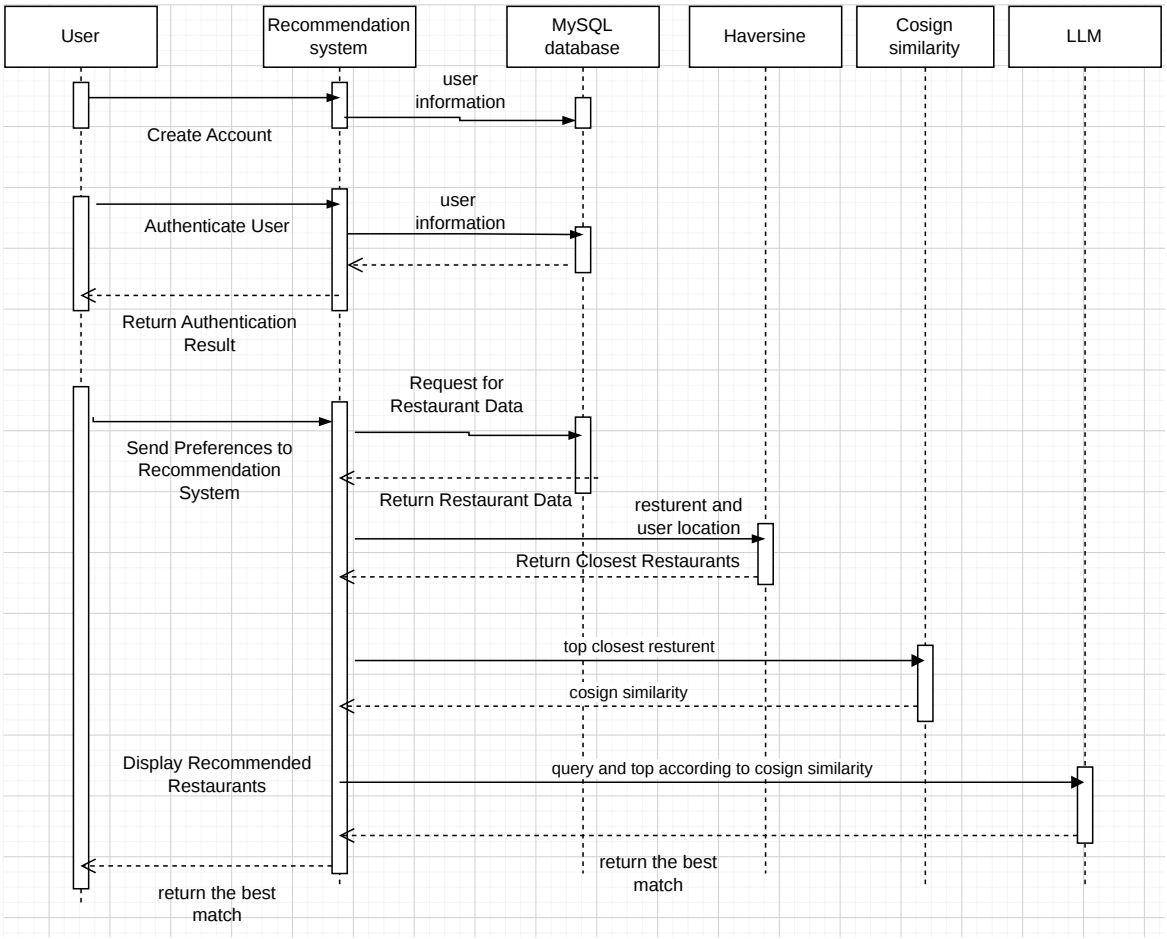

Figure 4.6: sequence diagram

- **Component Diagram :** This Component Diagram 4.7 illustrates the architecture of a recommendation system, showing how different components interact with each other.

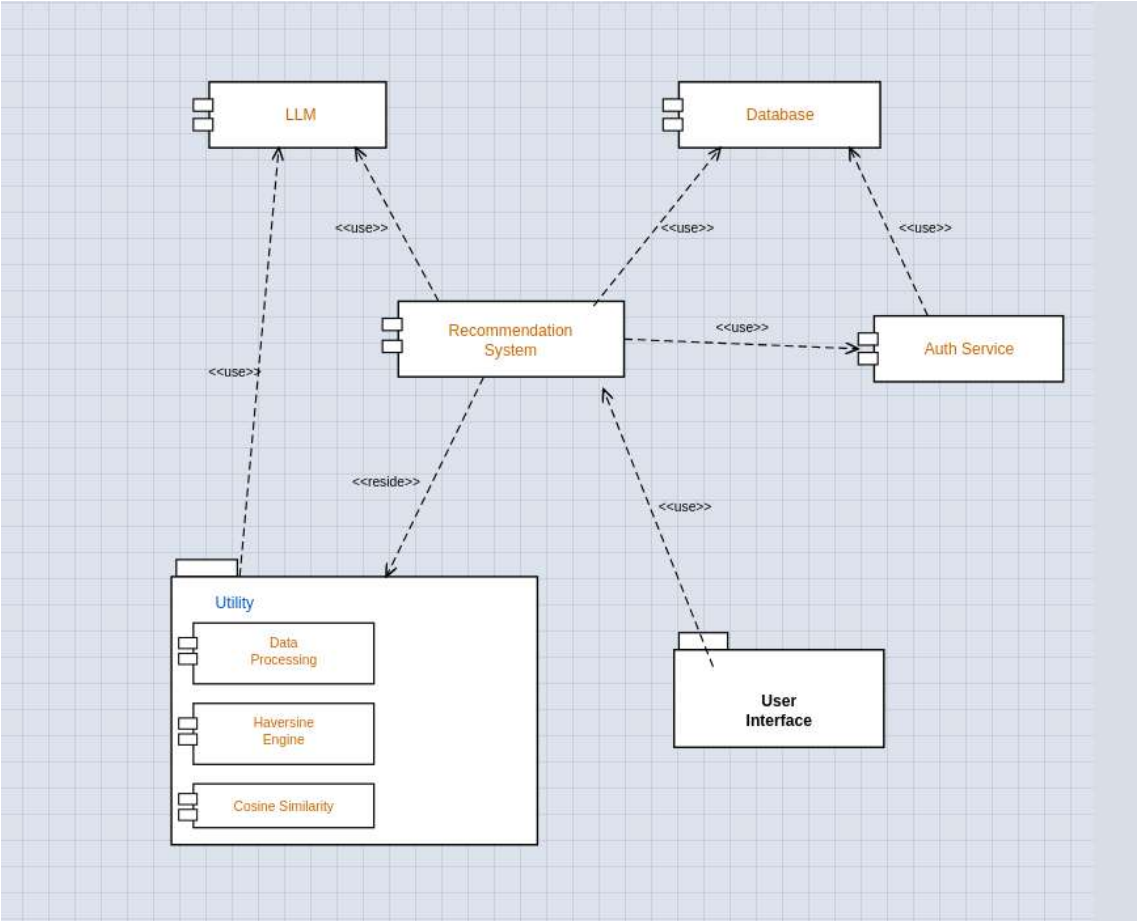

Figure 4.7: Component Diagram

- **Data Flow Diagram :** The interaction between user and the system begins with account creations and login where if a user is new to the system for getting his preferred recommendation he must create an account. After that the user provides his location and preference. Then using haversine formula and LLM model the system returned the best and closest restaurant that matched his preference. The data flow diagram illustrates how data moves from the user to the whole back and the LLM. The level zero diagram 4.8 is a high level view of data flow diagram which shows how the major components works and the level one 4.9 diagram more detail of the flow of the from the user input to the response.

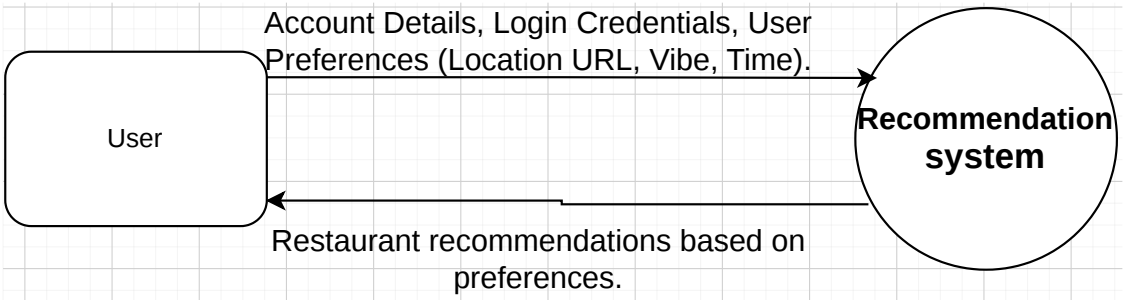

Figure 4.8: Context Data Flow Diagram

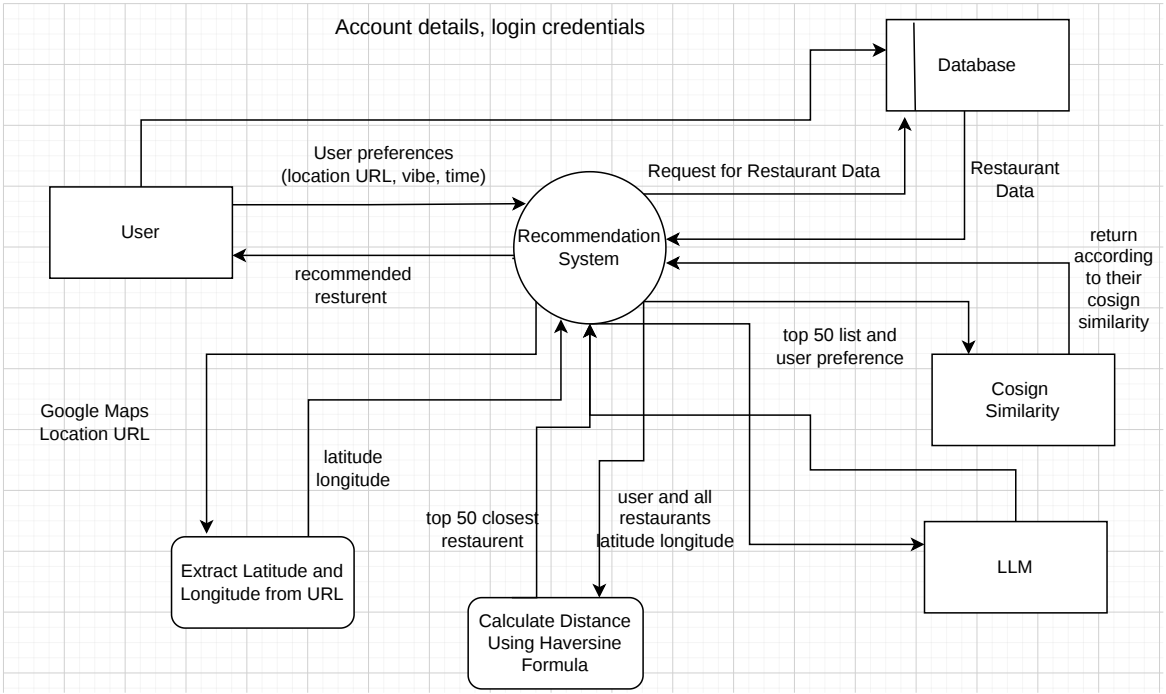

Figure 4.9: Level 0 Data Flow Diagram

- **Class Diagram :**

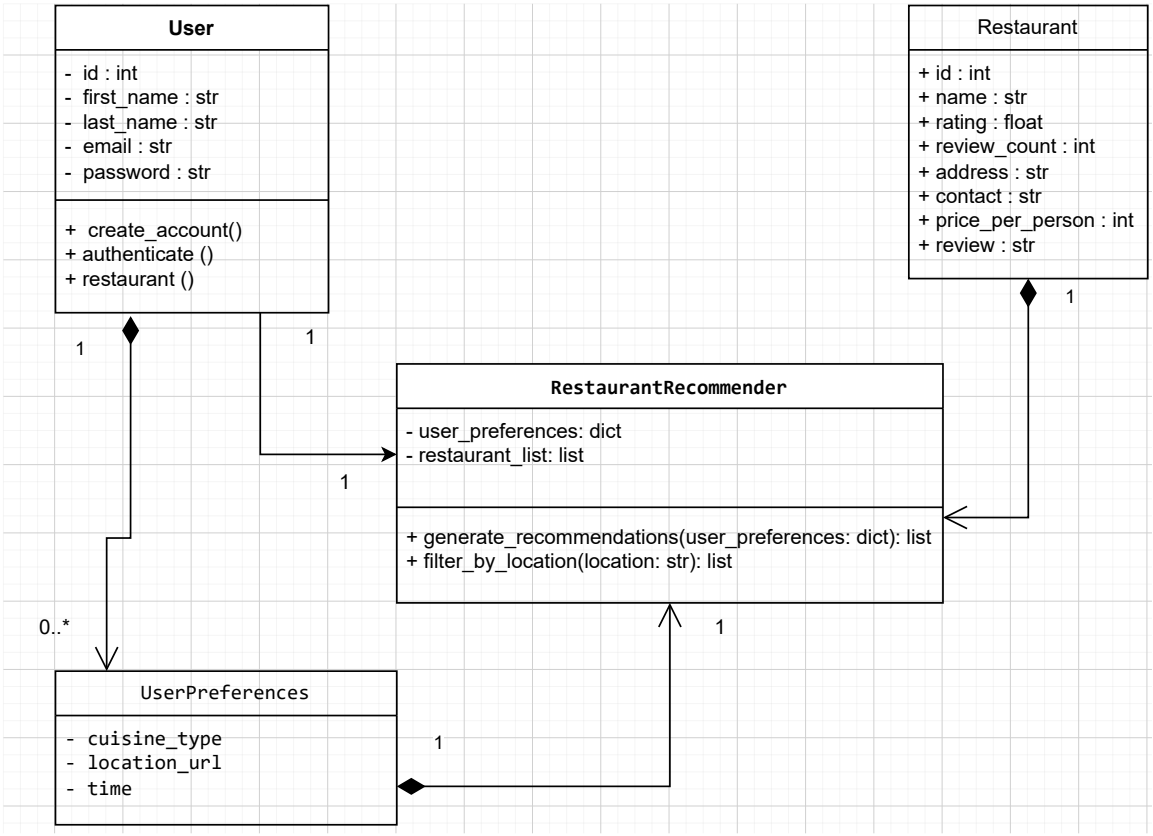

Figure 4.10: Class diagram

- **Components and Modules :** The system is built with several key components all of them are responsible for the specific task and all of these components work together to handle both frontend and backend. In the frontend part there is a homepage which is a landing page and when a user comes to the app the user initially comes to this page. Then there is a restaurant page where all the available restaurants are present. The user sees this restaurant without even login or creating an account to the platform. There is a login and create account page where the user login using his email and password and in the create account page user create account. After login, the user will visit his profile page where he can give his preference to get the preferred restaurant. The business logic and the data processing occur in the backend part. Though python and flask are being used for this project so there are several endpoints of the frontend part. For the home page (`/`) is used as the endpoint. For the find restaurant `/find_restaurant` as the endpoint here all the restaurant data react from the database and send to the front and for the user. For login and create account `/login` and `/create_account` endpoint is used here in the time of login the it verifies the data provided by the user and stored in the database and in terms of create account it verifies the data type and store it in the database. The system uses a mysql database for storing the data. There are two tables for the project. In the table 4.2 is the user table where the data is stored at the time of creating an account and fetching the data at the time of logging and generating recommendation. And the table 4.4 is stored all the

restaurant information that is used for the project.

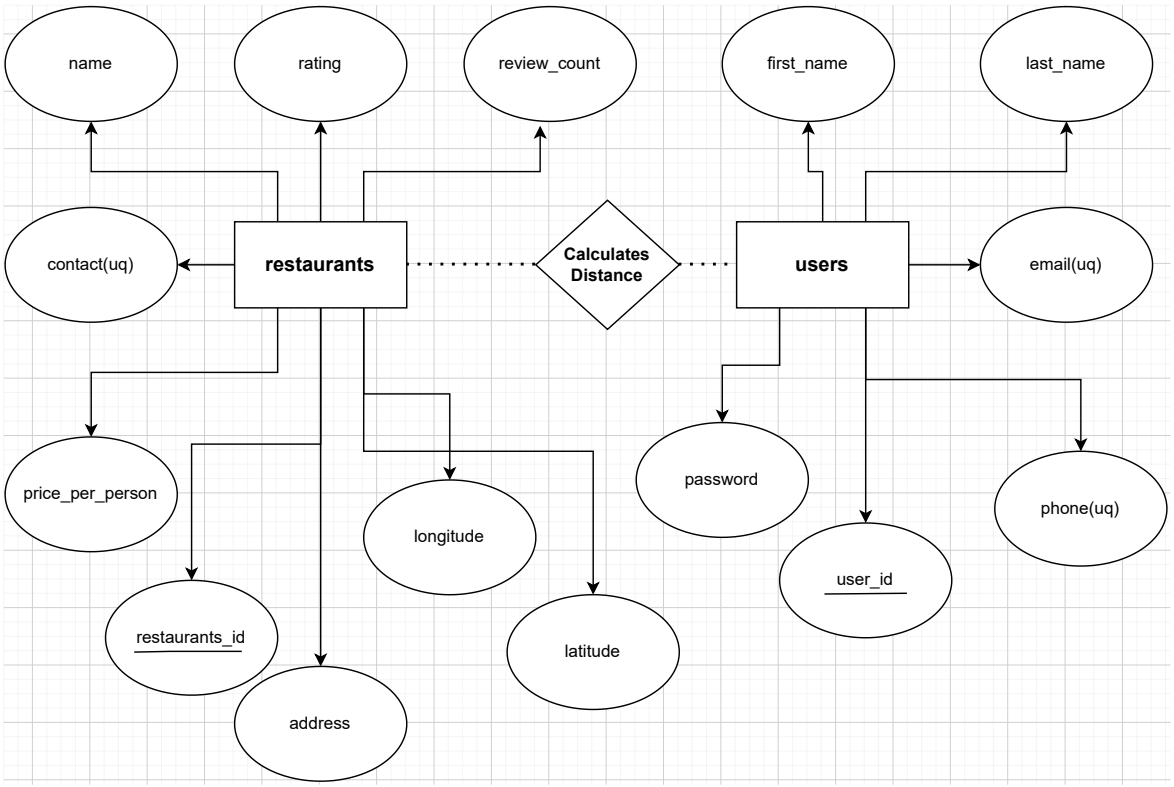

Figure 4.11: Entity-Relationship Diagram

Column	**Data Type**	**Constraints**	**Description**
user_id	INT	PK, AUTO_INCREMENT	Unique identifier for each user.
first_name	VARCHAR(50)	NOT NULL	User's first name.
last_name	VARCHAR(50)	NOT NULL	User's last name.
email	VARCHAR(100)	UNIQUE (uq), NOT NULL	Used for login and contact.
password	VARCHAR(255)	NOT NULL	Hashed password for security.
phone	VARCHAR(20)	UNIQUE (uq), NULL	Optional contact number.

Table 4.2: Data Dictionary for the `users` Entity

Column	Data Type	Constraints	Description
restaurants_id	INT	PK, AUTO_INCREMENT	Unique internal business ID.
name	VARCHAR(100)	NOT NULL	Name of the establishment.
address	TEXT	NOT NULL	Physical location string provided by the Google Maps API.
latitude	DECIMAL(10, 8)	NOT NULL	GPS Coordinate (Latitude) used for distance calculation.
longitude	DECIMAL(11, 8)	NOT NULL	GPS Coordinate (Longitude) used for distance calculation.
rating	DECIMAL(2, 1)	DEFAULT 0.0	Average star rating from scraped data.
review_count	INT	DEFAULT 0	Total number of user reviews recorded.
contact	VARCHAR(20)	UNIQUE (uq), NULL	Business phone number.
price_per_person	VARCHAR(20)	NULL	Estimated cost category for the establishment.

Table 4.4: Data Dictionary for the `restaurants` Entity

4.5.3 User Interaction and Interface

- **Homepage** After entering the website all the users primarily landed on this page.

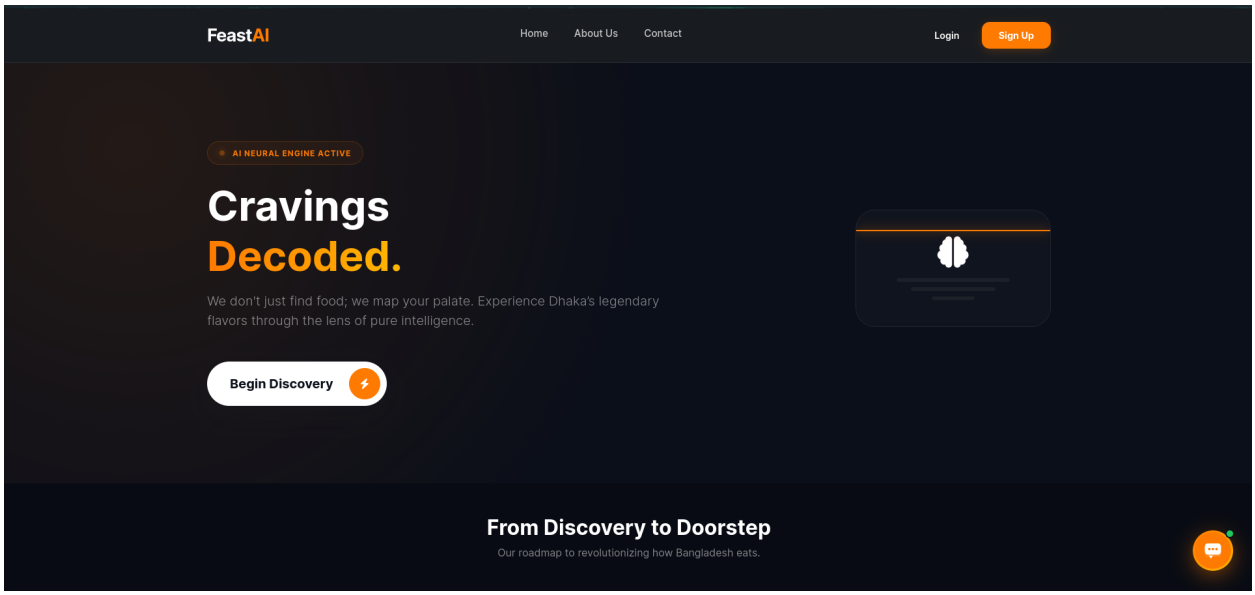

Figure 4.12: Homepage

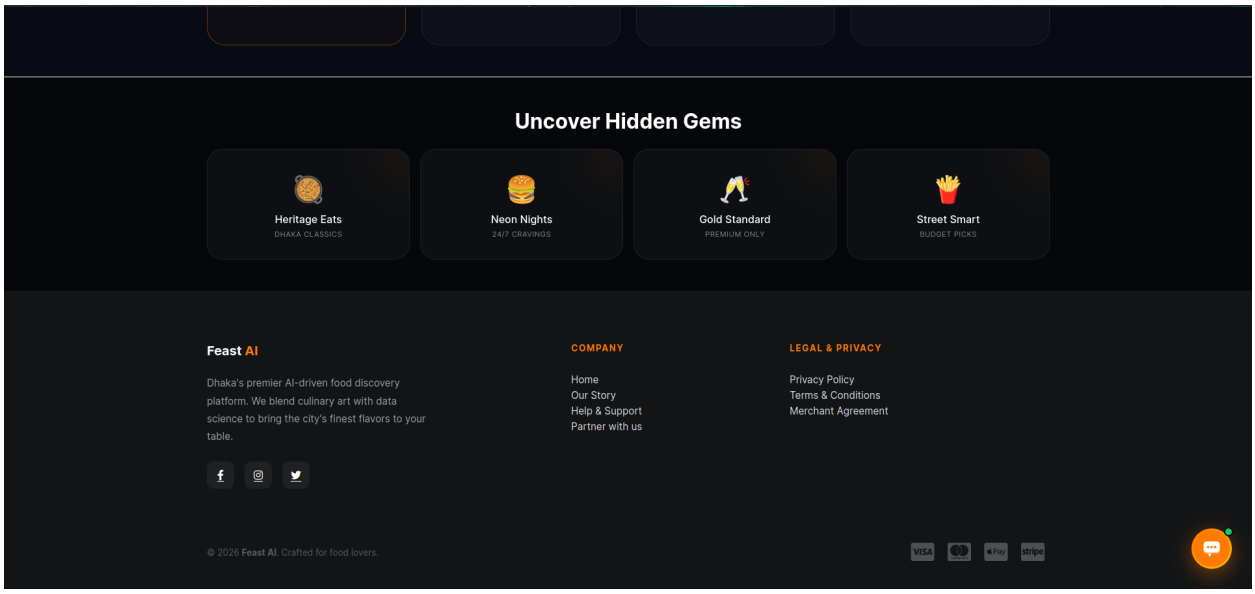

Figure 4.13: Homepage Footer

- **Chatbot** At the right corner of the page have a chatbot. Which allows the user to find his preference that is available in the system.

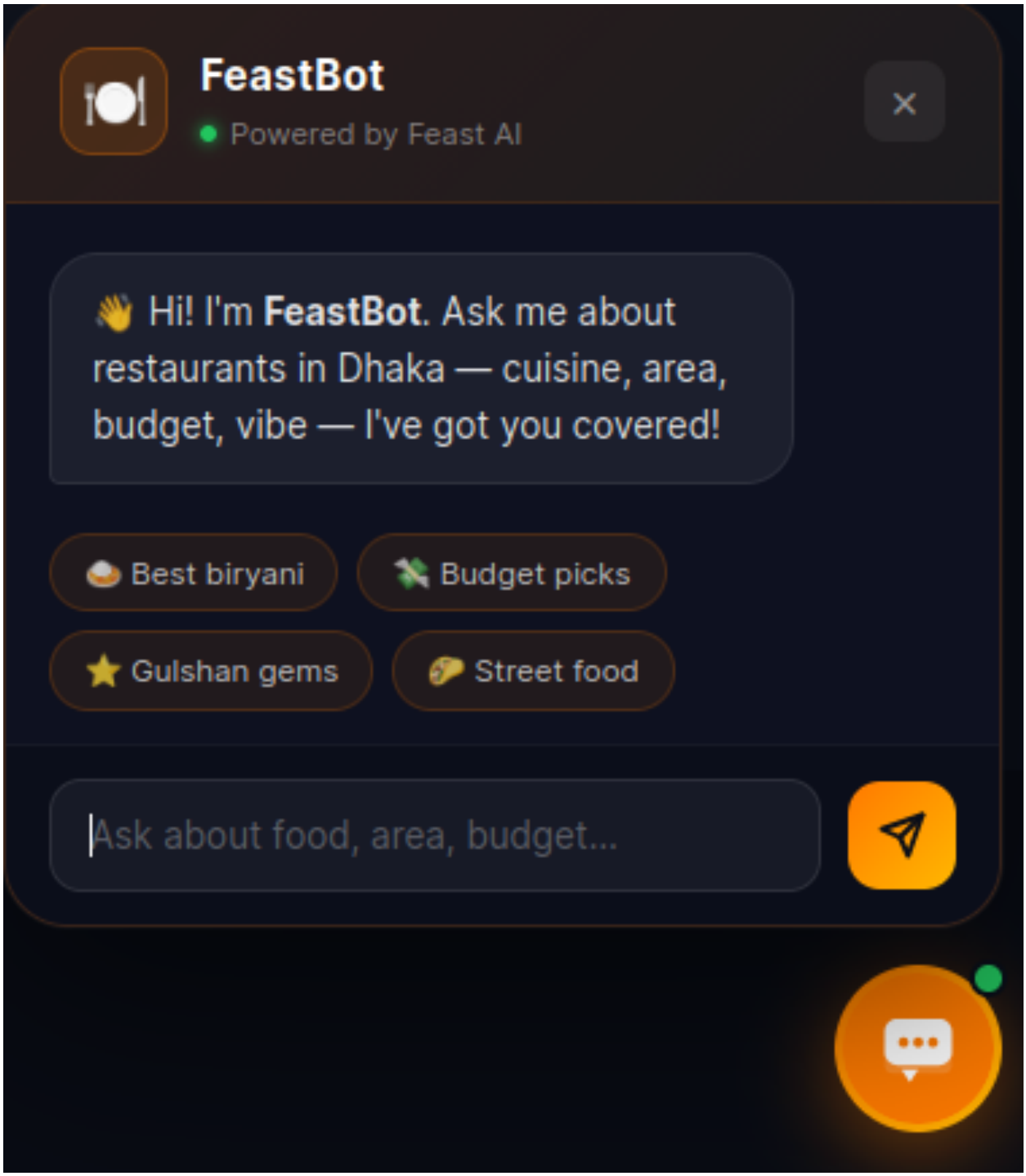

Figure 4.14: Chatbot

- **All Available Options** When the user clicks the "Begin Discovery" on the homepage it will bring the user to the all available restaurant page in the website. Here this page shows all the available restaurants in the system and their details.

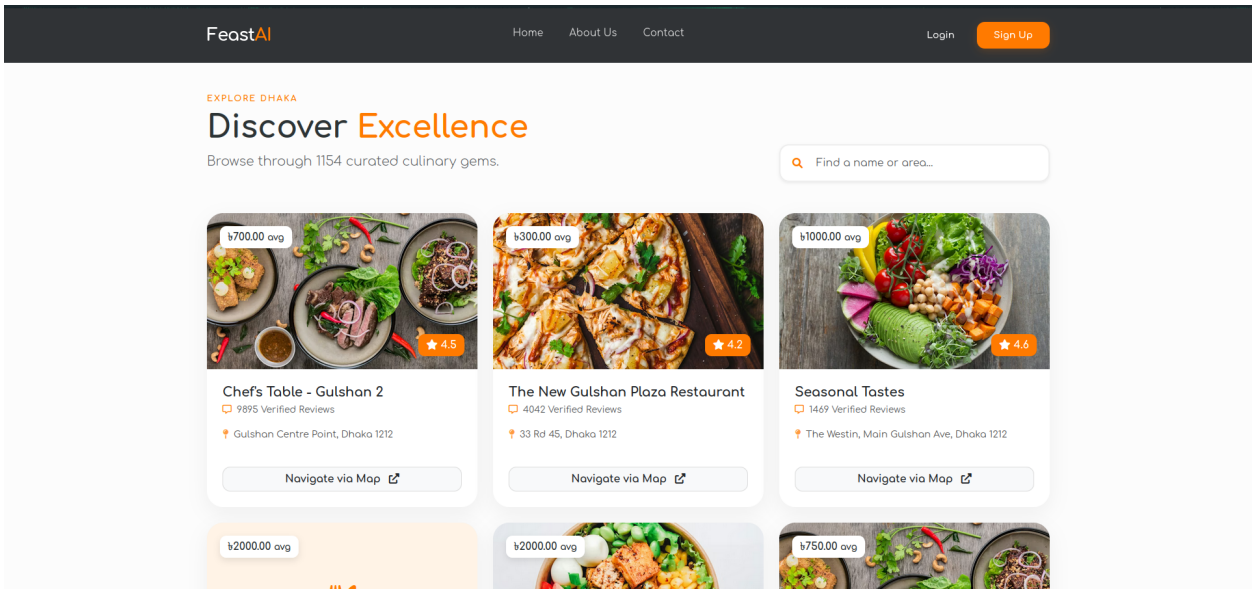

Figure 4.15: Available all restaurants

- **Login** When the user clicks on the login on the navbar it will bring the user to the login page. Every user can give their valid mail and password to login into the system. Which allows the user to get their personalized recommendations.

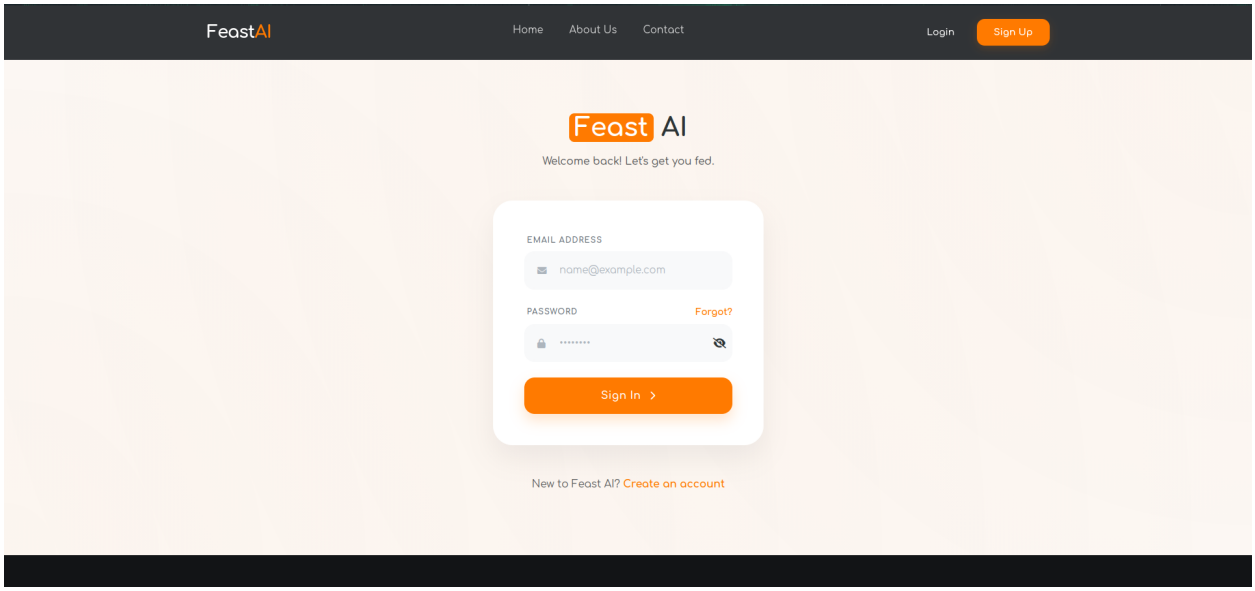

Figure 4.16: User Login

- **Create Account** When the user clicks on on the create account on the navbar it take the user to the create account page. Here a user can create an account by giving all the information.

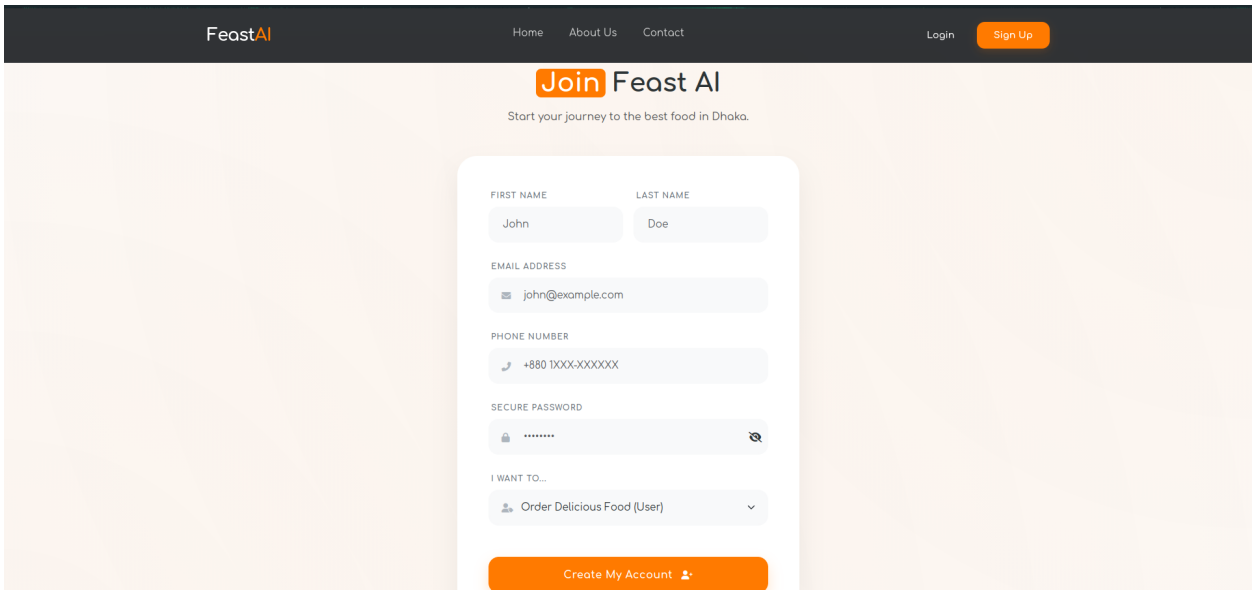

Figure 4.17: Create account

- **User Profile** After successful login the user landed on the profile page where the user saw his name, subscription type, and contact information.

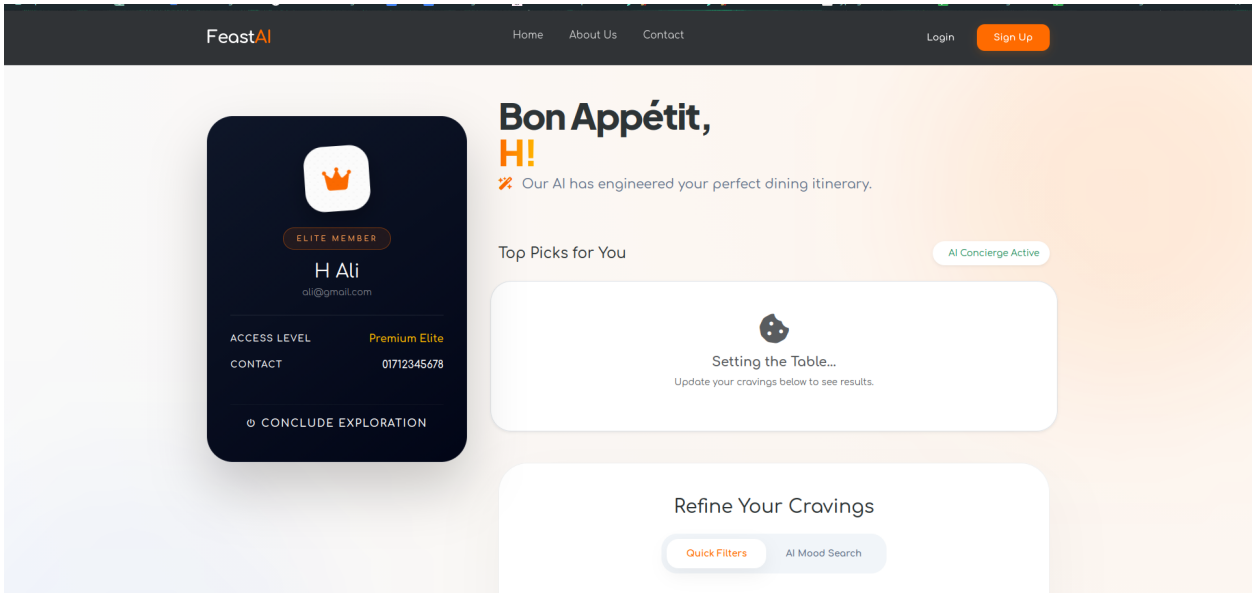

Figure 4.18: User Profile

- **Make Preference by Selecting** In this portion the user can give his preference and preferred time by selecting.

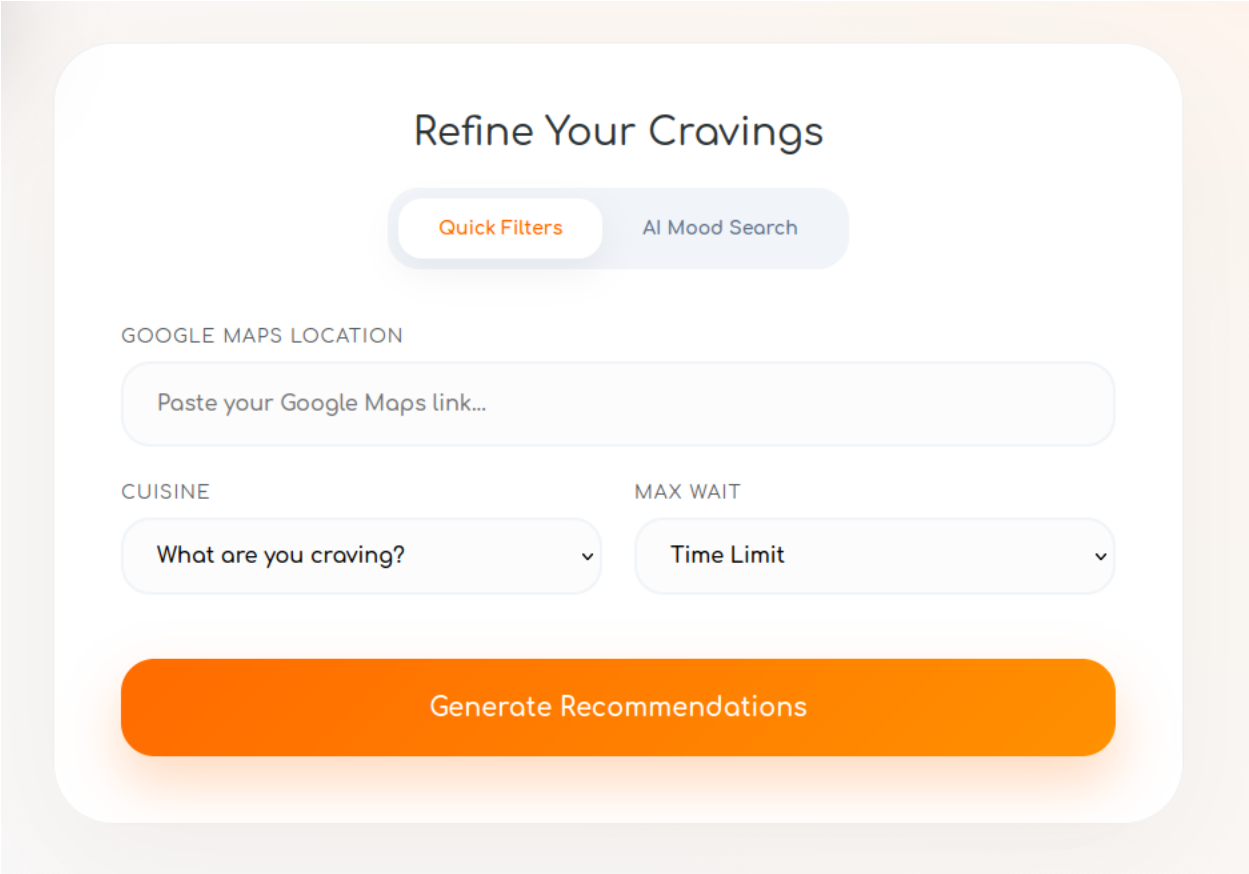

Figure 4.19: Make Preference by Selecting

- **Make Preference by Writing Text** Here the user can give his preference by writing textual format. Which gives users more freedom to express what he want.

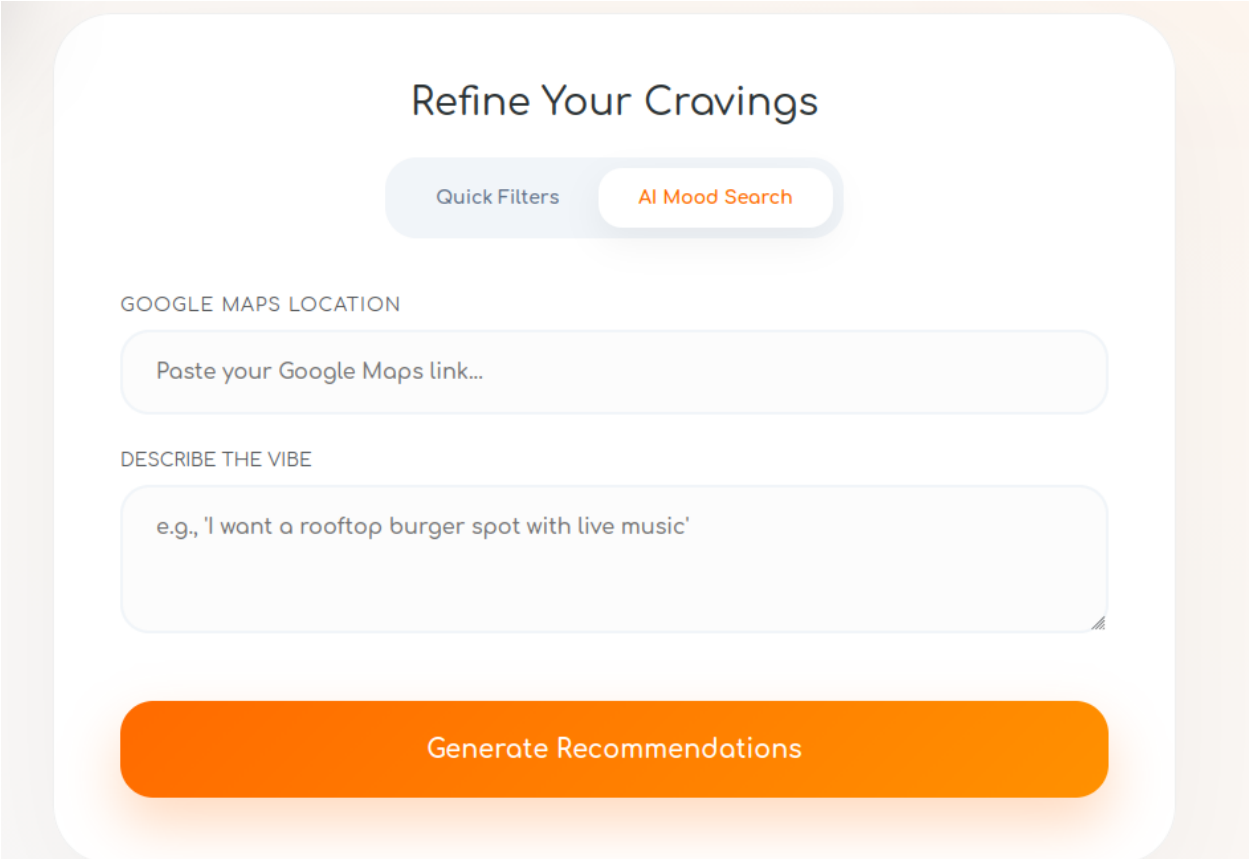

Figure 4.20: Make Preference by Writing Text

- **Getting Recommendations** After successfully providing preference and location and click generate recommendation the system generates the most suitable restaurant that matches the user vibe and preference.

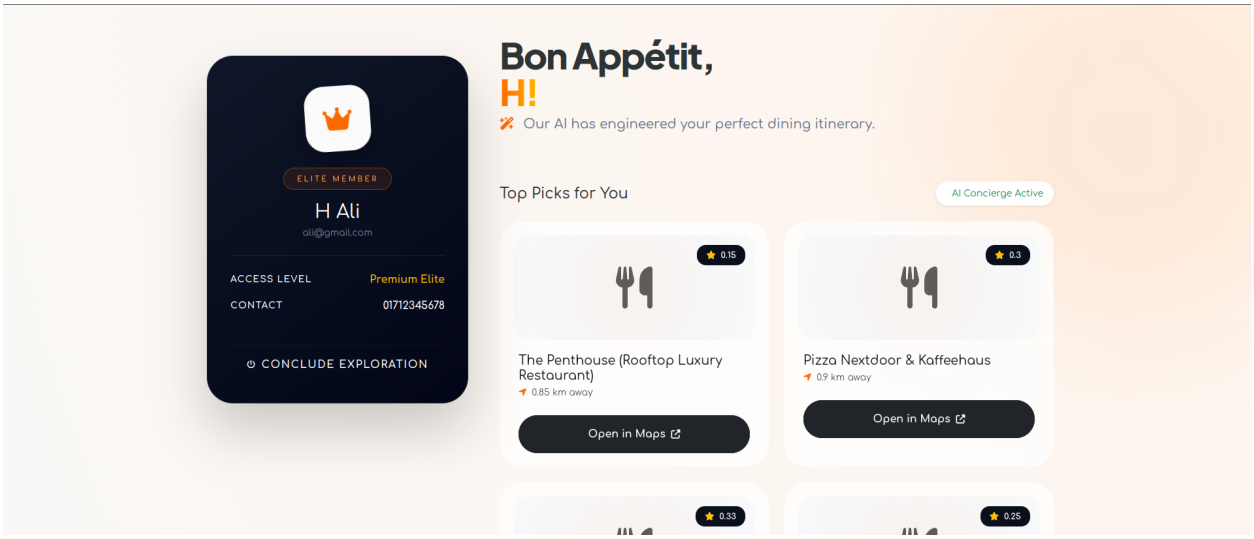

Figure 4.21: Getting Recommendations

4.5.4 Future Upgrades and Enhancements

- **Taste Matching:** currently a specially designed "Taste Score" Function is in the development pipeline that will propose items based on users specific taste, keeping into consider your individual tastes and previous suggestions. This will result in a more personalized dining experience compared to the current version while making sure that each meal is a great fit.

- **Mobile Launch:** The Feast AI system will soon be available on iOS as well as Android systems. This will allow our customers to effortlessly access the service we provide via the mobile app, making it even easier to find and order meals from their favorite restaurants.

- **National Reach Expansion:** Our one of the major priorities is to develop and expand our platform to include every major city in Bangladesh. This Expected implementation will make our platform more easily available, allowing us to have a national influence on how Bangladesh eats.

Chapter 5

Conclusion

Web automation tools such as BeautifulSoup help to extract the relevant information of the restaurant from the Google map. But for using this, there are some major issues IP lock. But using some smart trick, like a random time interval or vpn, ensures the smooth data extraction.

5.1 Summary of Findings

The research effectively showed that, even with the challenges of managing dynamic and JavaScript-rendered material, automated data collecting from Google Maps is possible. Important conclusions include: The effective data extraction is possible by the use of BeautifulSoup and Selenium. Collecting all the relevant and important information is easier. IP lock, dynamic class names, and sporadic missing data were among the difficulties the project encountered. These problems were successfully resolved by using techniques like creating random time intervals between requests and rotating IP addresses using VPN. Following extraction, the data was cleaned by eliminating inconsistencies, duplicates, and missing values, ensuring that the dataset was correct and ready for analysis.

5.2 Contributions to the Field

This effort adds to the expanding corpus of research on automation and online scraping methods, especially as they relate to location-based data extraction. Among the contributions are: Using Selenium and BeautifulSoup, a scalable technique for extracting substantial quantities of structured data from Google Maps is demonstrated. The information gathered may be used in several domains, including sentiment analysis, location-based services, urban planning, and market research. Researchers and practitioners can get useful insights from the strategies used to overcome issues like IP blocking and dynamic class names when scraping data from websites with comparable restrictions. The project makes a new standard of customer service by providing a low-cost, high-quality alternative to existing platforms and demonstrating how free technology can be used in real-world applications by proper utilization.

5.3 Recommendations for Future Work

Even though the project is implementing its preliminary version, there are lots of scoops to improve it such as: The standard and resilience of the error handling process might be improved by implementing more options of handling network error, web pages missing and so on. Investigating Google Maps APIs for data extraction may provide a more organized and dependable way to get location-based data while abiding by the platform's terms of service and lowering the possibility of IP restrictions. Applications that need current information, such as tracking company evaluations or ratings, would benefit from integrating live data streams and using real-time data extraction. To get a greater understanding of location-based company performance, future research may also use sophisticated data analysis approaches, including machine learning models, to forecast ratings or trends based on the data gathered.

Bibliography

[1] M. Chemlal, A. Zedadra, O. Zedadra, and M. N. Kouahla, "A personalized restaurant recommendation system using ml-topsis approach," in *International Conference on Emerging Intelligent Systems for Sustainable Development (ICEIS 2024)*, Atlantis Press, 2024, pp. 270–285.

[2] J. Tian, Z. Wang, J. Zhao, and Z. Ding, "Mmrec: Llm based multi-modal recommender system," in *2024 19th International Workshop on Semantic and Social Media Adaptation & Personalization (SMAP)*, 2024, pp. 105–110. DOI: 10.1109/SMAP63474.2024.00028

[3] W. M. Lim, R. Agarwal, A. Mishra, and A. Mehrotra, "The rise of fake reviews: Toward a marketing-oriented framework for understanding fake reviews," *Australasian Marketing Journal*, vol. 33, no. 2, pp. 178–198, 2025. DOI: 10.1177/14413582241283505 eprint: https://doi.org/10.1177/14413582241283505. [Online]. Available: https://doi.org/10.1177/14413582241283505

[4] J. Lin et al., "How can recommender systems benefit from large language models: A survey," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–47, 2025.

[5] Y. S. Soekamto, A. Lim, L. C. Limanjaya, Y. K. Purwanto, S.-H. Lee, and D.-K. Kang, "Pic2plate: A vision-language and retrieval-augmented framework for personalized recipe recommendations," *Sensors*, vol. 25, no. 2, 2025, ISSN: 1424-8220. DOI: 10.3390/s25020449 [Online]. Available: https://www.mdpi.com/1424-8220/25/2/449

[6] A. Ubaid, A. Lie, and X. Lin, "Smart restaurant recommender: A context-aware restaurant recommendation engine," *AI*, vol. 6, no. 4, 2025, ISSN: 2673-2688. DOI: 10.3390/ai6040064 [Online]. Available: https://www.mdpi.com/2673-2688/6/4/64