Restaurant Recommendation System

Team ID:-592805

Index Page

1. INTRODUCTION

1.1 Project Overview:-

A restaurant recommendation system (RRS) is the focus of this research. An information filtering system called a recommendation system makes an attempt to forecast the rating a user would assign to the item—in this case, a restaurant. RRS is an online restaurant search system. All Bangalore's restaurants are available for browsing. Obtain details about the name, kind, rating, and cost of the restaurant. Among the features are restaurant searches and recommendation viewing. Systems that provide recommendations are essential for boosting sales for businesses and enabling customers to locate eateries that suit their preferences. The fact that so many users don't rate the restaurants and users that are introduced to the system every day makes it difficult to use. We must forecast the rating for the restaurants that are not rated in order to enhance the restaurant rating system. Therefore, developing a recommendation system for restaurants with low ratings is crucial. Users only need to enter the name of a restaurant they have enjoyed visiting in the past into this recommendation algorithm, and it will produce a list of the top 10 restaurants based on the highest cosine similarity scores to that specific restaurant. In order to offer restaurants that match a user's interests, the content-based recommendation methodology suggests restaurants to consumers based on related restaurant categories and popular topic keywords.

1.2 Purpose

This system's goal is to give consumers suggestions for restaurants that would be best for them. People can acquire suggestions from this method, and you can also get other people's viewpoints via this website. Additionally, you can browse the ratings page, which compiles feedback and experiences from numerous individuals, to identify the top eateries. The idea behind this system is that users may browse through the data you send and find all the restaurants that have responded to consumers' requests.

This system functions similarly to a bulletin board for foodies. Customers must use this website to look up restaurants by name. They will get a page describing the related names of the restaurants and their type and ratings.

2. LITERATURE SURVEY

2.1 Existing problem

As we are users of recommendation applications, people care more about how we will like a restaurant. It is very common that we hang out with families, friends, and co-workers. when comes to lunch or dinner time. In the past, people obtained suggestions for restaurants from friends. Although this method is straightforward and user-friendly, it has some severe limitations. First, the recommendations from friends or other common people are limited to those places they have visited before. Thus, the user is not able to gain information about places less visited by their friends. Besides that, there is a chance of users not liking the place recommended by their friends. So our application will try to solve this problem.

2.2 References

https://medium.com/mlearning-ai/restaurant-recommendation-system-based-on-the-content-in-reviews-dfc3351004db

1)supervised and unsupervised learning

Link: https://www.youtube.com/watch?v=kE5QZ8G_78c

2)Regression, Classification and Clustering

Link: https://www.youtube.com/watch?v=6za9_mh3uTE

3)ML-content based recommender System

Link: https://www.geeksforgeeks.org/ml-content-based-recommender-system/

4)NLTK: Natural Language tool kit

Link:- https://www.nltk.org/

5)Flask

Link: https://www.youtube.com/watch?v=lj4I_CvBnt0

6)Recommendation System

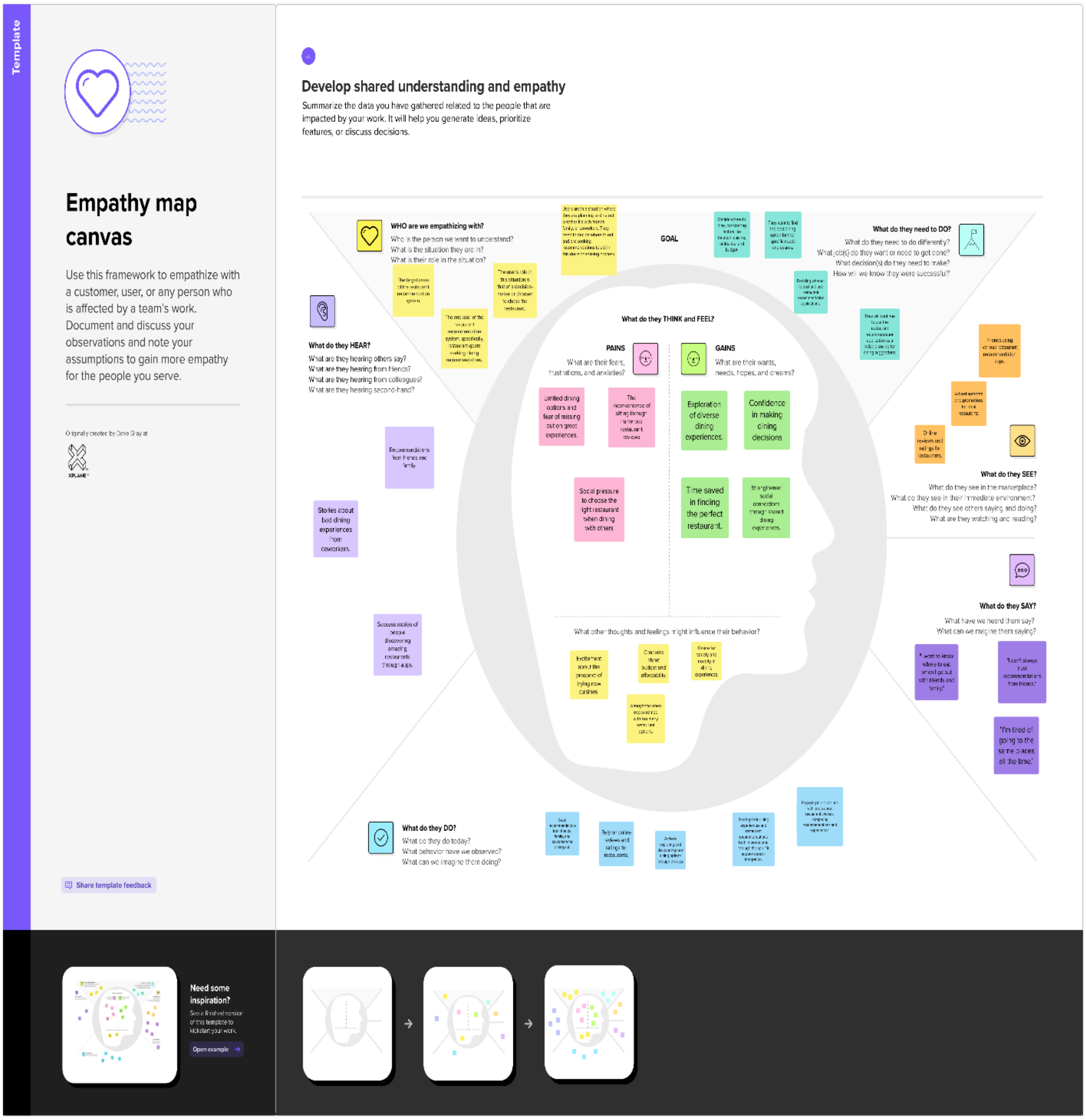
Link: https://www.youtube.com/watch?v=n3RKsY2H-NE

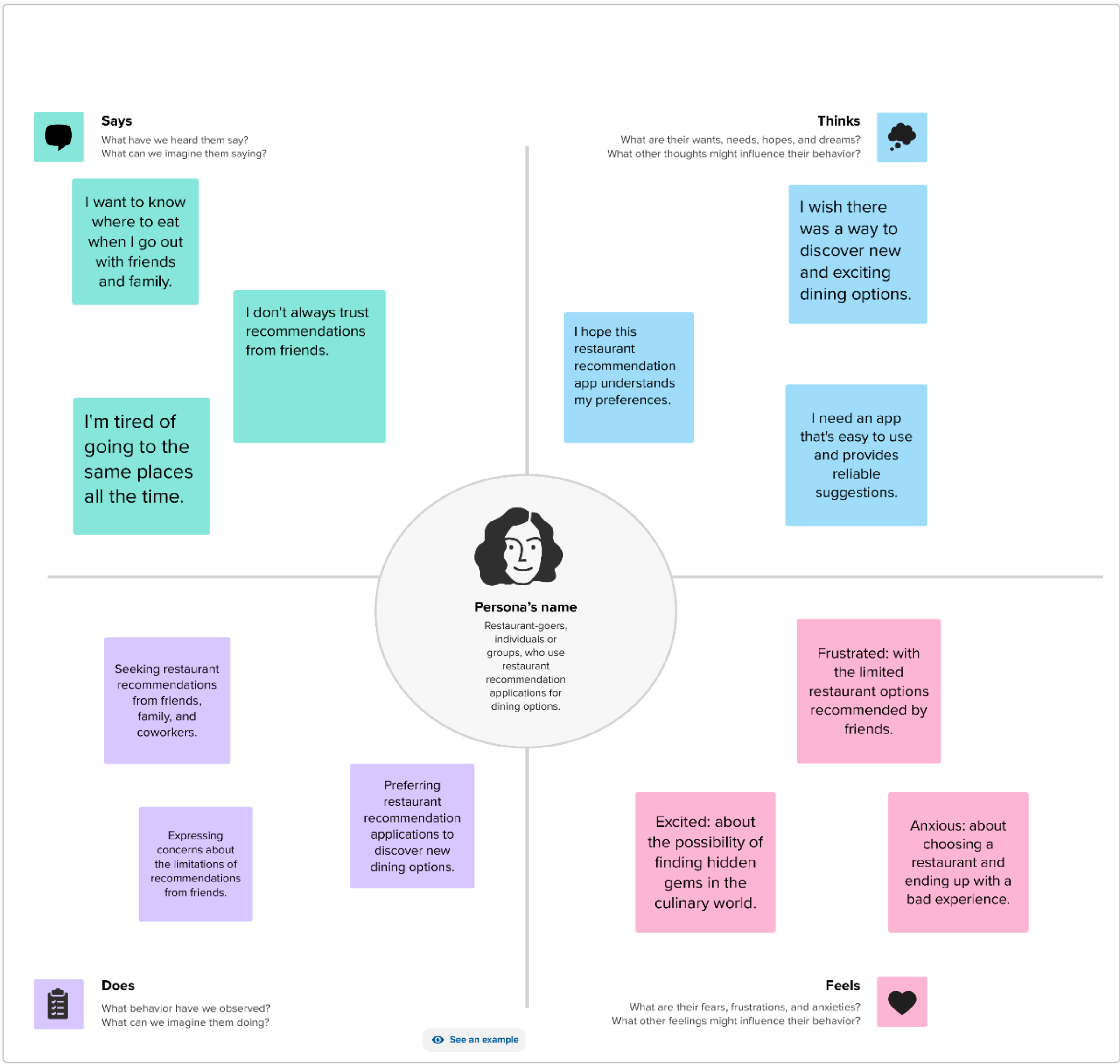
2.3 Problem Statement Definition

As we are users of recommendation applications, people care more about how we will like a restaurant. It is very common that we hang out with families, friends, and co-workers. when comes to lunch or dinner time. In the past, people obtained suggestions for restaurants from friends. Although this method is straightforward and user-friendly, it has some severe limitations. First, the recommendations from friends or other common people are limited to those places they have visited before. Thus, the user is not able to gain information about places less visited by their friends. Besides that, there is a chance of users not liking the place recommended by their friends. So our project gives a way to user to find similar restaurants to the restaurants they already like without asking for suggestions from their friends or family.

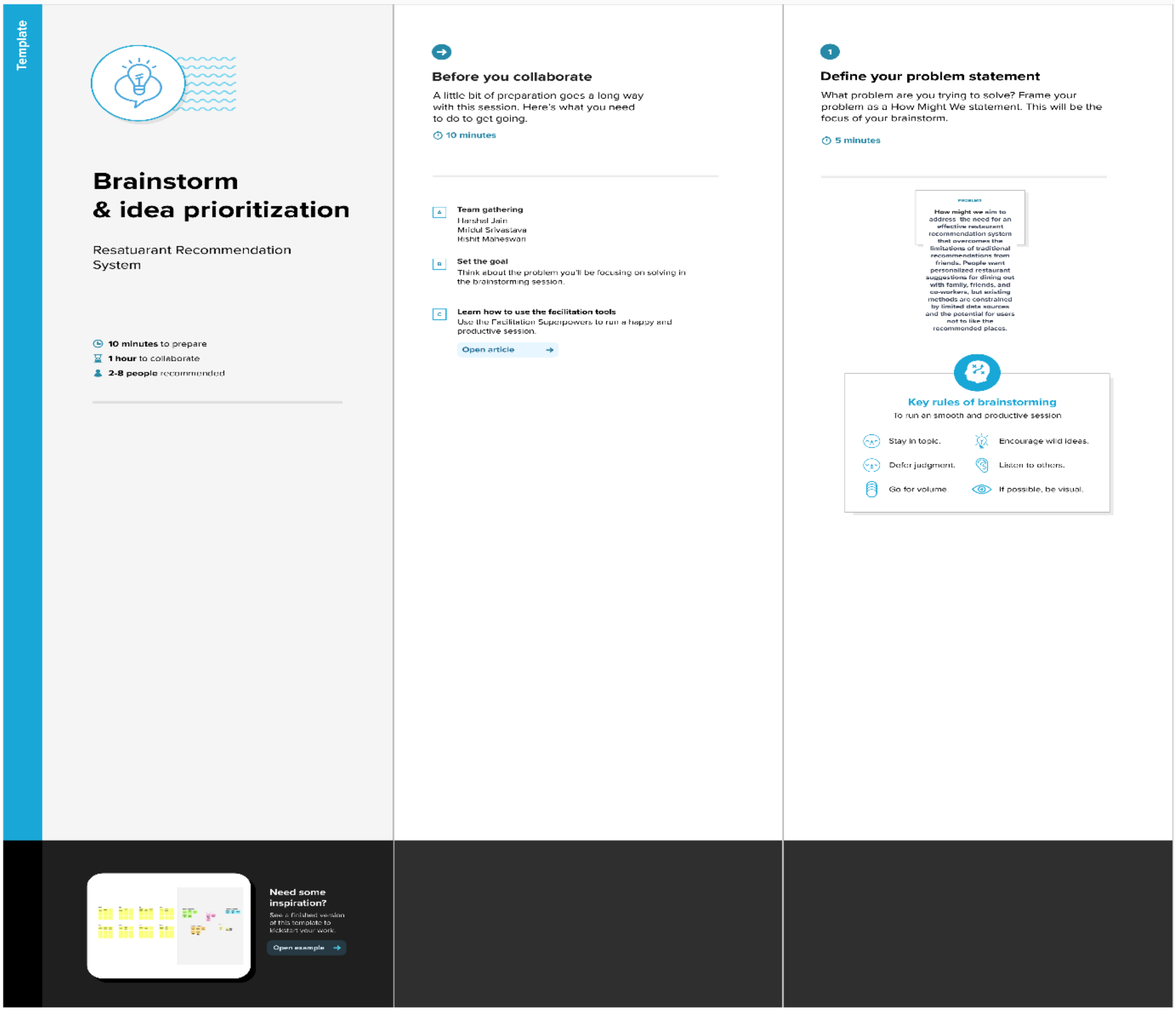
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas





3.2 Ideation and Brainstorming



PROBLEM

How might we aim to address the need for an effective restaurant recommendation system that overcomes the limitations of traditional recommendations from friends. People want personalized restaurant suggestions for dining out with family, friends, and co-workers, but existing methods are constrained by limited data sources and the potential for users not to like the recommended places.

2

Brainstorm

Write down any ideas that come to mind
that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Harshal

- Web App
- can use user location
- Can Connect to zomato App

Rishit

- No. of users
- Can consider rating
- Can use existing dataset

Mridul

- mapbox
- affordability
- Rating System

- Web App
- Can use existing dataset
- Can consider rating
- Rating System
- mapbox

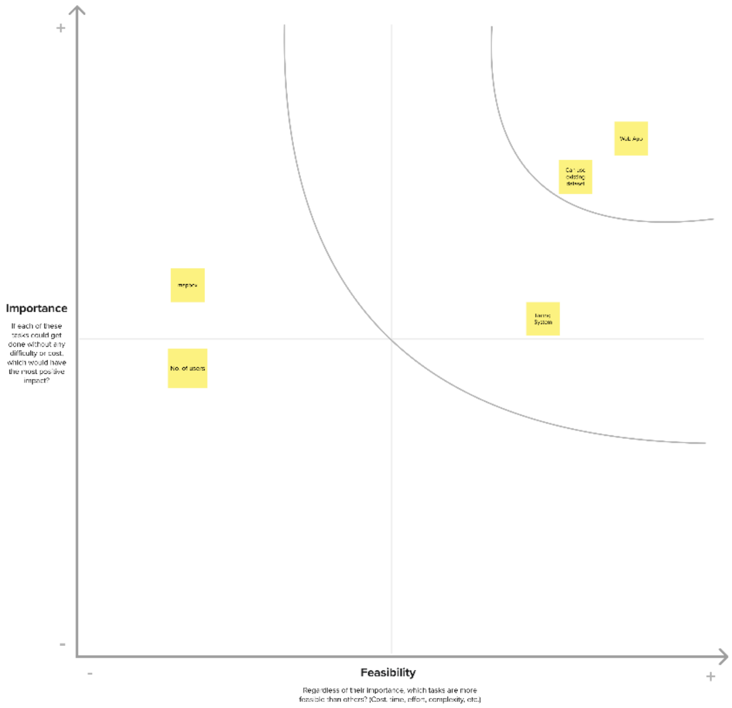
Prioritize

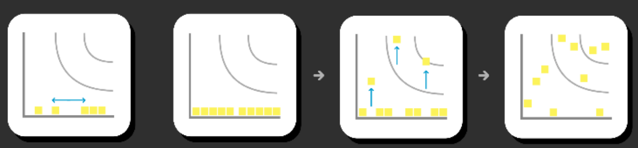
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.





After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

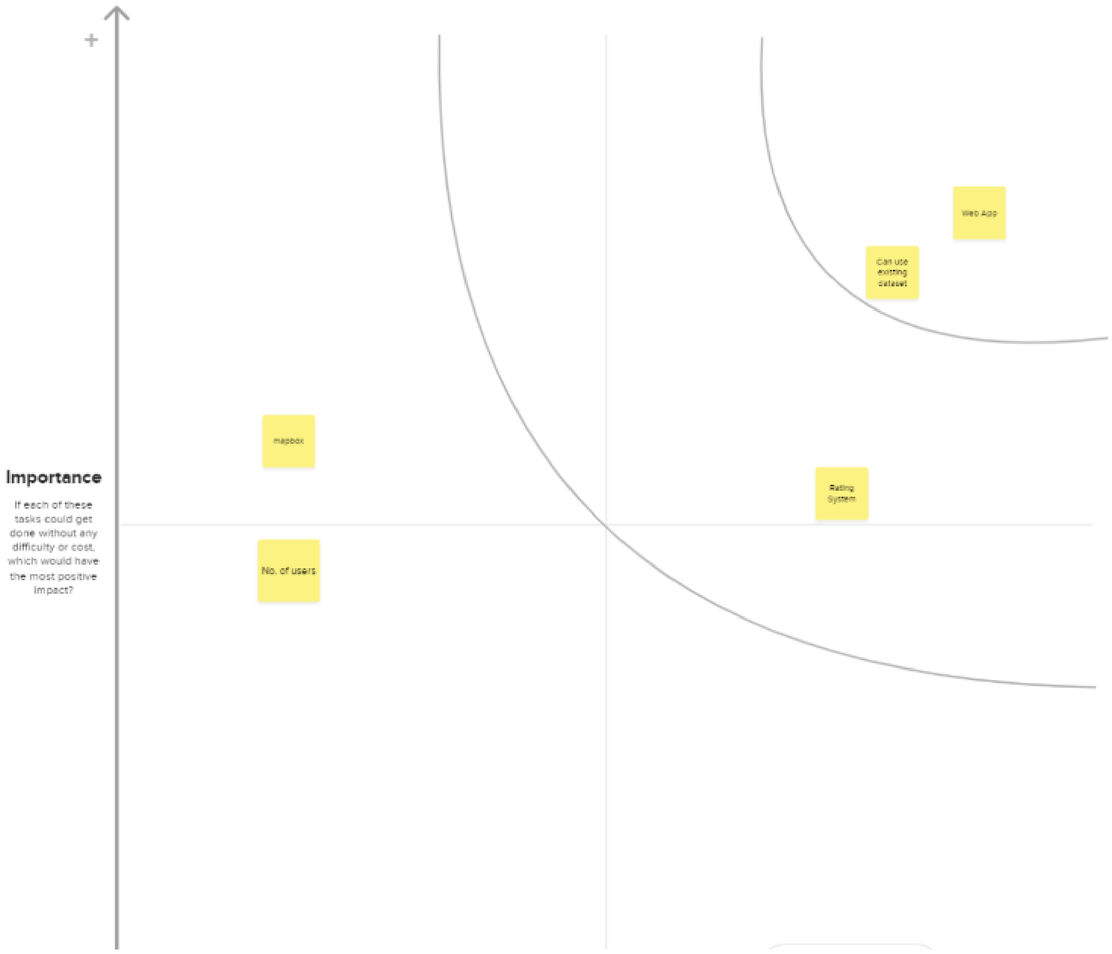
Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback



4. REQUIREMENT ANALYSIS

4.1 Functional Requirement

Hardware and Software Software Requirements:

To complete this project, you will require the following software's, concepts, and packages

Anaconda navigator Python packages:

- pandas
- matplotlib
- seaborn
- plotly
- numpy
- scikit-image
- scikit-learn
- Flask

4.2 Non-Functional Requirements

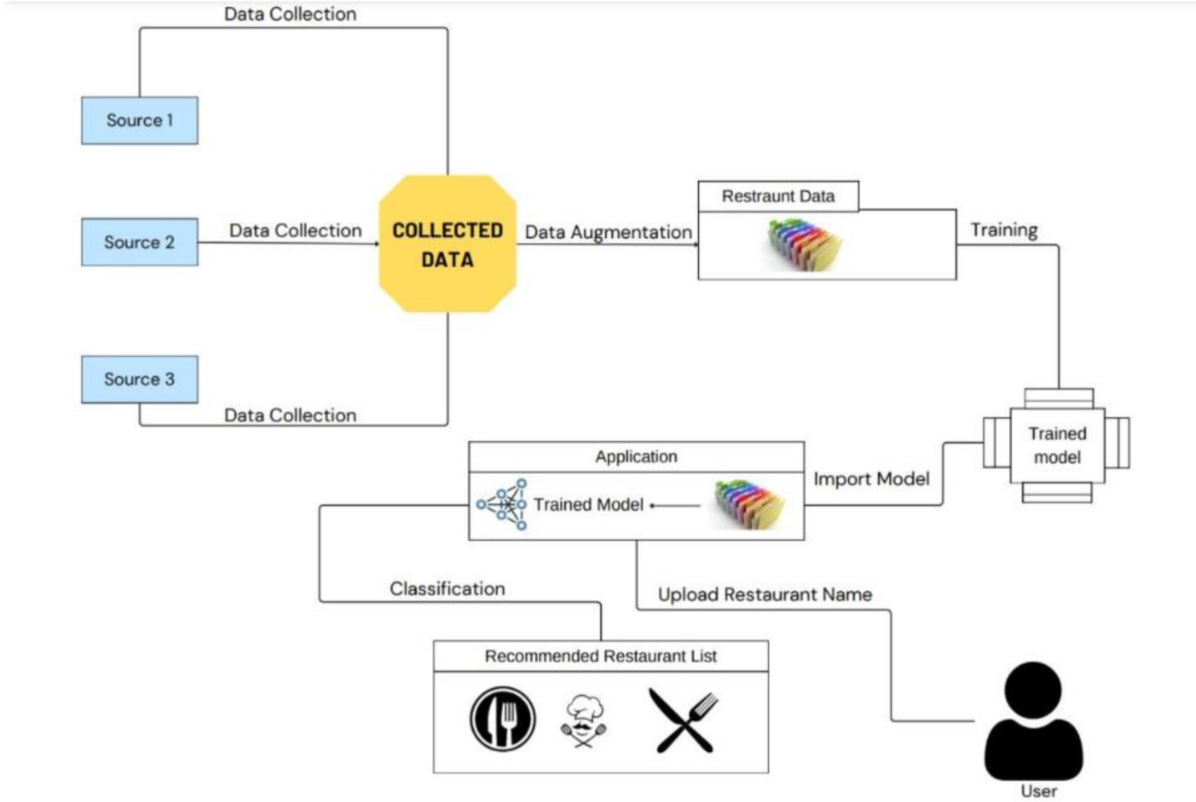
Hardware Requirements

- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 4 GB

Other than this we will also need the user to enter a restaurant according to which our recommendation system will make the recommendations.

5. PROJECT DESIGN

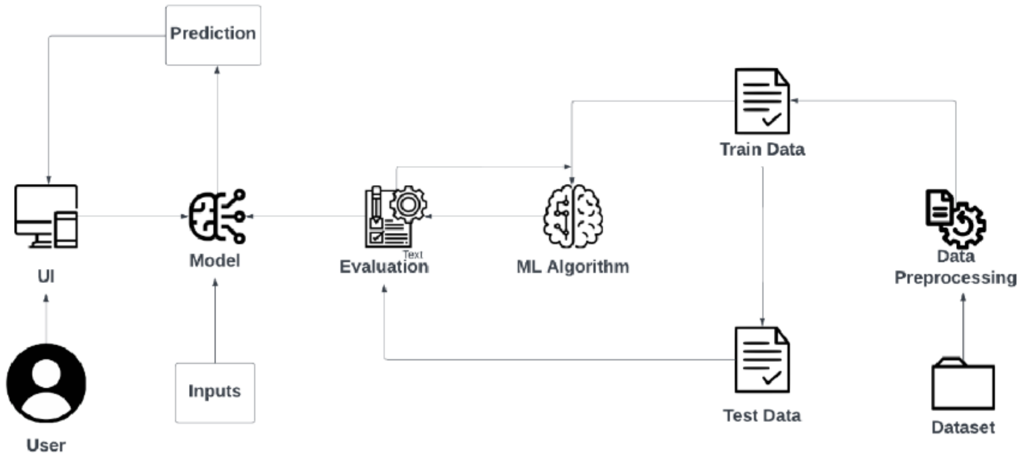
5.1 Data Flow Diagrams and User Stories



User Stories	Functional requirement(Epic)	User Story Number	User Story/Task	Acceptance criteria	Priority	Release
Get restaurant recommendations	Provide personalized restaurant recommendations based on the customer's preferences, such as cuisine, location, price range, and rating.	US1	As a customer, I want to be able to get personalized restaurant recommendations so that I can find the best places to eat.	The system must be able to return a list of restaurants that match the customer's preferences, including the cuisine, location, price range, and rating.	High	1.0
Filter restaurant recommendations	Allow customers to filter restaurant recommendations by various criteria, such as ratings and cuisine.	US2	As a customer, I want to be able to filter restaurant recommendations by various criteria so that I can find the perfect restaurant for my needs.	The system must allow customers to filter restaurant recommendations by the following criteria: rating, review and cuisine.	Medium	1.1
Share restaurant recommendations with friends.	Allow customers to share restaurant recommendations with their friends.	US3	As a customer, I want to be able to share restaurant recommendations with my friends so that I can help them find the best places to eat.	The system must allow customers to share restaurant recommendations with their friends via email, social media, or other messaging platforms.	Low	1.2

5.2 Solution Architecture

Solution Architecture Diagram:-



Workflow:

- User inputs a restaurant name via the web interface.
- The Flask app processes the input and interacts with the recommendation model.
- Text data from the Zomato dataset undergoes TF-IDF vectorization.
- Cosine similarity calculation to find similar restaurants based on reviews.
- Top similar restaurants are selected and displayed to the user via the web interface.

Technologies Used:

- Python (Flask, Pandas, NumPy, Seaborn, Matplotlib, Plotly, Scikit-learn)
- Pickle for model serialization
- Web development: HTML/CSS, Jinja templating
- NLP libraries: NLTK for text processing

This architecture supports a user-friendly interface where users can discover similar restaurants based on their preferences and explores a solution using text-based analysis for recommendations.

6.1 Technical Architecture:

Overview:

The technical architecture of the restaurant recommendation system includes various components:

- **Frontend (User Interface):**
    - Developed using HTML/CSS, with Flask framework for rendering dynamic content.
- **Backend (Application Logic):**
    - Flask application interacting with a recommendation model.
    - Utilizes Pandas, NumPy, NLTK, and Scikit-learn for data processing and recommendation generation..
- **Machine Learning Model (Pickle File):**
    - A serialized model used to generate restaurant recommendations.

Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI	HTML, CSS
3.	Database	Collect the Dataset Based on the Problem Statement	Kaggle
4.	File Storage/ Data	File storage requirements for Storing the dataset	Local System, Google Drive
5.	Frame Work	Used to Create a web Application, Integrating Frontend and Back End	Python Flask
6.	Deep Learning Model	Purpose of Model	ML algorithms, K-nearest algorithm
7	7. Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud	Local, AWS

6.2 Sprint Planning and Estimation:-

Sprint	Functional Requirement(Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Project Setup and infrastructure	USN-1	Setting up the development environment and tools and required framework to start the project	1	High	Harshal
Sprint-1	Development Environment	USN-2	Explore the development environment to understands its capacity	2	High	Harshal
Sprint-2	Data Collection	USN-3	Gathering a diverse dataset to train our model	2	High	Rishit
Sprint-2	Data Pre-processing	USN-4	Pre-process the dataset to make it clean by removing null values or unnecessary columns	3	High	Harshal
Sprint-3	Model Development	USN-5	Evaluate different ML algorithm/Deep learning architecture to create a suitable model	3	high	Mridul
Sprint-3	Training	USN-6	Train the model using the pre-processed dataset	3	medium	Rishit
Sprint-4	Model Deployment	USN-7	Deploy the model as a web page for that write the html and css for the web page	3	High	Mridul
Sprint-4	Integration	USN-8	Integrate the model to the web page using Flask.	2	High	Harshal
Sprint-5	Testing	USN-9	Test you completed application	2	Medium	Rishit

6.3 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed(as on planned end date)	Sprint Release Date(Actual)
Sprint-1	3	1 day	23^{rd} oct	23^{rd} oct	3	23^{rd} oct
Sprint-2	5	1 day	24^{th} oct	24^{th} oct	8	24^{th} oct
Sprint-3	6	2 days	25^{th} oct	26^{th} oct	14	26^{th} oct
Sprint-4	5	3 days	25^{th} oct	27^{th} oct	19	27^{th} oct
Sprint-5	2	1 day	28^{th} oct	29^{th} oct	21	30^{th} oct

7.1 Feature 1: Improved Recommendation System

Feature 1:

Code and Description: The code performs data preprocessing and cleaning. It drops unnecessary columns, handles missing values, computes mean ratings, and processes text data for analysis.

```python
# Drop unnecessary columns
zomato_df = zomato_df.drop(['phone','dish_liked'], axis=1)

# Remove NaN values
zomato_df.dropna(how='any', inplace=True)

# Drop duplicate rows
zomato_df.drop_duplicates(inplace=True)

# Rename columns
zomato_df = zomato_df.rename(columns={
    'approx_cost(for two people)': 'cost',
    'listed_in(type)': 'type',
    'listed_in(city)': 'city'
})

# Clean 'rate' column
# Processing rates and converting them to float
zomato_df = zomato_df.loc[zomato_df.rate != 'NEW']
zomato_df = zomato_df.loc[zomato_df.rate != '-'].reset_index(drop=True)
remove_slash = lambda x: x.replace('/5', '') if type(x) == np.str else x
zomato_df.rate = zomato_df.rate.apply(remove_slash).str.strip().astype('float')

# Processing 'cost' column
zomato_df['cost'] = zomato_df['cost'].astype(str)
zomato_df['cost'] = zomato_df['cost'].apply(lambda x: x.replace(',', '.'))
zomato_df['cost'] = zomato_df['cost'].astype(float)

# Compute Mean Rating
restaurants = list(zomato_df['name'].unique())
zomato_df['Mean Rating'] = 0
for i in range(len(restaurants)):
    zomato_df['Mean Rating'][zomato_df['name'] == restaurants[i]] = zomato_df['rate'][zomato_df['name'] == restaurants[i]].mean()
# Scaling the mean rating values
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(1, 5))
zomato_df[['Mean Rating']] = scaler.fit_transform(zomato_df[['Mean Rating']]).round(2)
```

Feature 2:

Code and Description: The code snippet performs text processing for a restaurant recommendation system using TF-IDF and cosine similarity to generate top similar restaurants based on user input.

```python
# Text Processing for Recommendation System
# Creating tf-idf matrix
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'])

cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

# Recommendation Function
def recommend(name, cosine_similarities=cosine_similarities):
    # Creates a list to put top restaurants
    recommend_restaurant = []

    # Find the index of the input restaurant name
    idx = indices[indices == name].index[0]

    # Find the restaurants with similar cosine-sim value and order them
    score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False)

    # Extract top 30 restaurant indexes with similar cosine-sim value
    top30_indexes = list(score_series.iloc[0:31].index)

    # Names of the top 30 restaurants
    for each in top30_indexes:
        recommend_restaurant.append(list(df_percent.index)[each])

    # Return a DataFrame with similar restaurant details
    return df_new
```

Database Schema:

Based on the processed data, the hypothetical schema might be represented as follows:

Table: restaurants

- `url` (VARCHAR)
- `name` (VARCHAR)
- `online_order` (VARCHAR)
- `book_table` (VARCHAR)
- `rate` (FLOAT)
- `location` (VARCHAR)
- `cuisines` (VARCHAR)
- `cost` (FLOAT)
- `reviews_list` (VARCHAR)
- `city` (VARCHAR)
- `Mean Rating` (FLOAT)

This schema might correspond to the DataFrame `zomato_df` used within the code for data analysis.

8.1 Performance Metrics:

1. **Response Time:**
    - **Definition:** The time taken for the system to respond to a user query.
    - **Metric:** Average response time for a given number of concurrent users.
2. **Throughput:**
    - **Definition:** The number of requests processed within a given time frame.
    - **Metric:** Requests per second (RPS) or transactions per second.
3. **Scalability:**
    - **Definition:** The ability of the system to handle increased load or users without significant performance degradation.
    - **Metric:** Response time and throughput as the number of concurrent users increases.
4. **Concurrency and Load Testing:**
    - **Definition:** Evaluating system performance under various concurrent user loads.
    - **Metric:** Response time and system stability under increasing user load.
5. **Error Rates:**
    - **Definition:** Frequency of errors or failed requests under varying loads.
    - **Metric:** Error rate as a percentage of total requests.
6. **Resource Utilization:**
    - **Definition:** Analysis of system resources like CPU, memory, and network utilization during testing.
    - **Metric:** CPU and memory consumption during peak loads.
7. **Cache Hit Ratio (if applicable):**
    - **Definition:** The ratio of cache hits to cache lookups, indicating cache efficiency.
    - **Metric:** Percentage of cache hits.

Testing Methodologies:

1. **Load Testing:**
    - Simulate a realistic load on the system using tools like JMeter, locust.io, or custom scripts to assess system behavior under normal and peak loads.
2. **Stress Testing:**
    - Apply loads beyond normal operational capacity to identify the breaking point or system failure point.
3. **Endurance Testing:**
    - Sustain a load for an extended period to detect memory leaks or performance degradation over time.
4. **Soak Testing:**

- Subject the system to a sustained load for an extended period to evaluate its stability and performance under prolonged stress.

5. **Scalability Testing:**
   - Measure the system's performance as resources are added or as the user load increases, aiming to identify performance bottlenecks.

Evaluation Criteria:

- **Acceptable Response Time:** Define a baseline for response time that would be considered acceptable for users (e.g., < 2 seconds).
- **Scalability Limits:** Identify the maximum number of concurrent users the system can handle without drastic performance degradation.
- **Resource Utilization Thresholds:** Determine acceptable levels of CPU, memory, and network usage under varying loads.
- **Error Thresholds:** Define an acceptable error rate (e.g., < 1%) for the system.

By assessing these performance metrics and testing methodologies, the system can be evaluated for its responsiveness, stability, and capacity to handle user loads, ensuring a smooth and efficient experience for users seeking restaurant recommendations.

```
recommend('Cinnamon')
```

TOP 10 RESTAURANTS LIKE Cinnamon WITH SIMILAR REVIEWS:

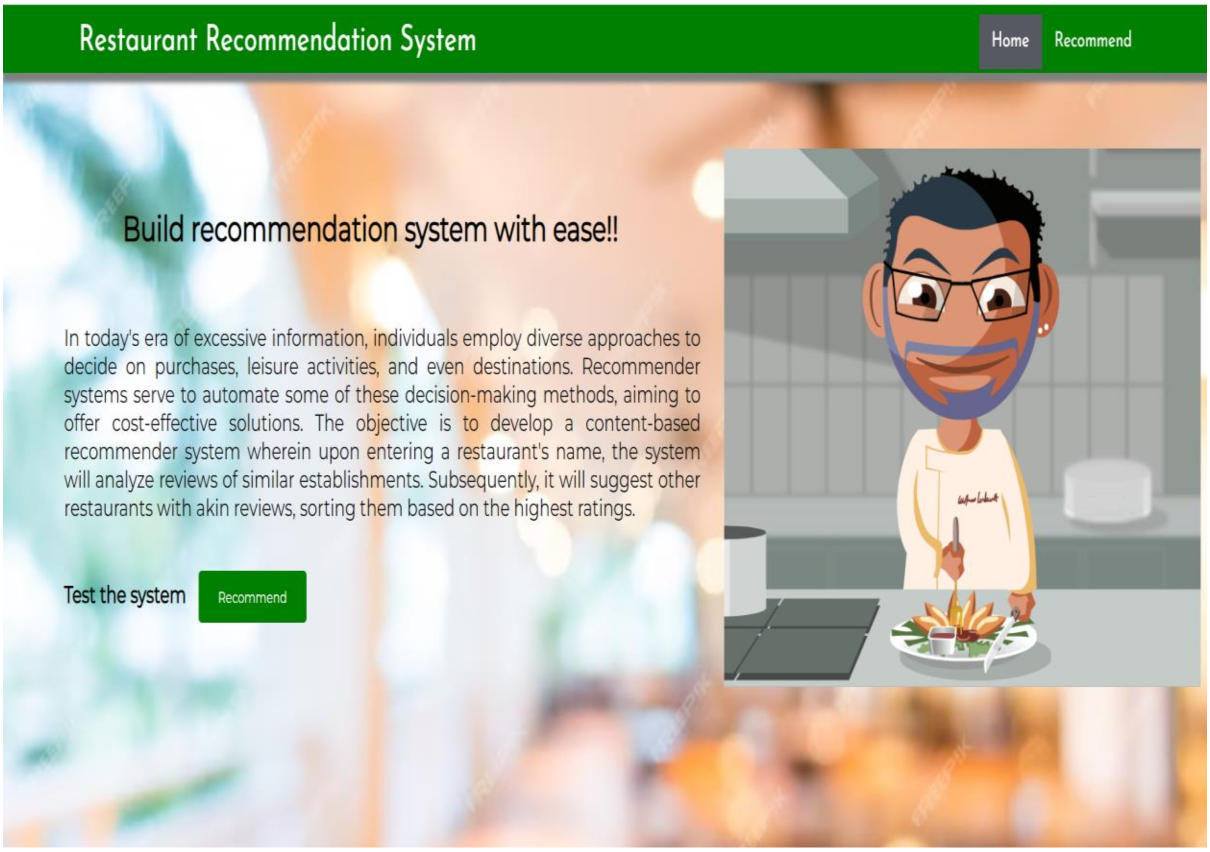
	cuisines	Mean Rating	cost	url
Eggzotic	North Indian, Chinese, Biryani, Fast Food	3.77	500.0	https://www.zomato.com/bangalore/eggzotic-doml...
Eggzotic	North Indian, Chinese, Biryani, Fast Food	3.77	500.0	https://www.zomato.com/bangalore/eggzotic-doml...
Cinnamon	North Indian, Chinese, Biryani	3.62	550.0	https://www.zomato.com/bangalore/cinnamon-hsr?...
Cinnamon	North Indian, Asian, Continental	3.62	1.0	https://www.zomato.com/bangalore/cinnamon-sesh...
Cinnamon	North Indian, Chinese, Biryani	3.62	550.0	https://www.zomato.com/bangalore/cinnamon-hsr?...
Cinnamon	North Indian, Chinese, Biryani	3.62	550.0	https://www.zomato.com/bangalore/cinnamon-hsr?...
Altaf's Chillies Restaurant	North Indian, Chinese	3.61	500.0	https://www.zomato.com/bangalore/altafs-chilli...
Donne Biriyani Angadi Mane	Biryani, Chinese	3.47	250.0	https://www.zomato.com/bangalore/donne-biriyan...
Tasty Paradise	North Indian, Mughlai, Chinese, Beverages	3.45	550.0	https://www.zomato.com/bangalore/tasty-paradis...
Dhaba Express	Biryani, Fast Food, North Indian, Chinese	3.45	400.0	https://www.zomato.com/bangalore/dhaba-express...

```python
# Define two restaurant names for similarity test
item1 = "Eggzotic"
item2 = "Cinnamon"

# Find the index of the restaurant entered
idx1 = zomato_df[zomato_df['name'] == item1].index[0]
idx2 = zomato_df[zomato_df['name'] == item2].index[0]

# Calculate the similarity score between the two restaurants
similarity_score = cosine_similarities[idx1, idx2]
print(f"Similarity Score between {item1} and {item2}: {similarity_score}")
```

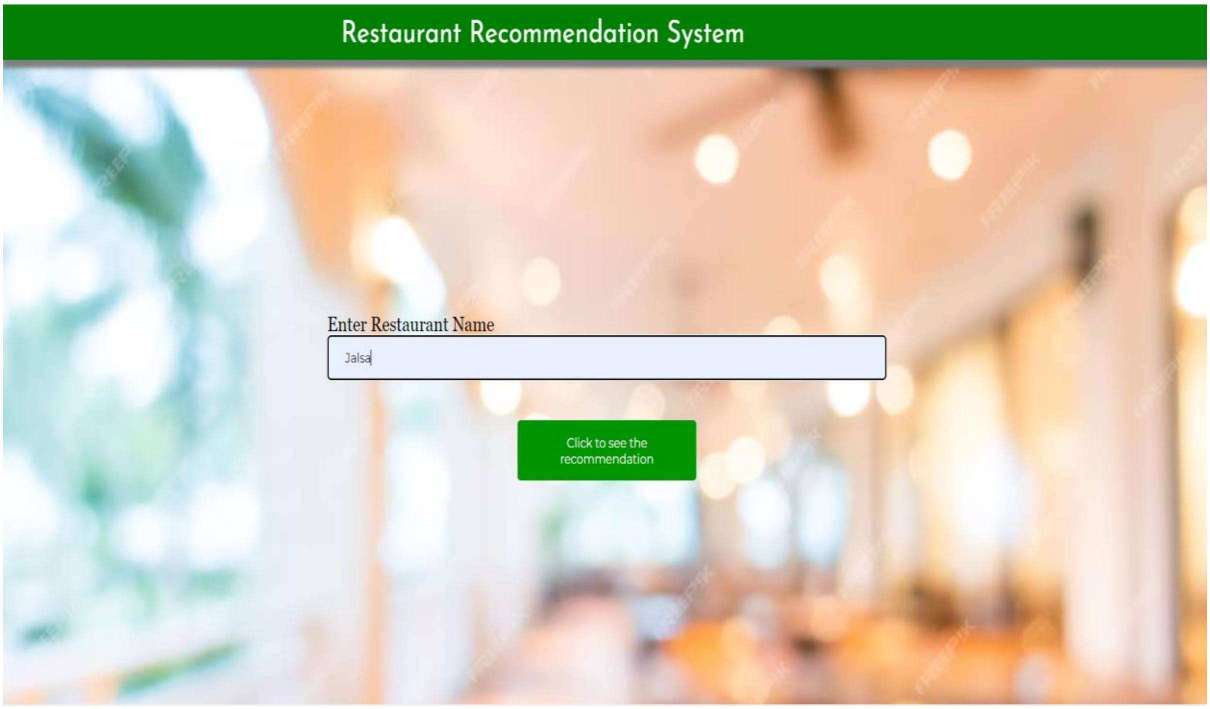
```
Similarity Score between Eggzotic and Cinnamon: 0.942595719800211755
```

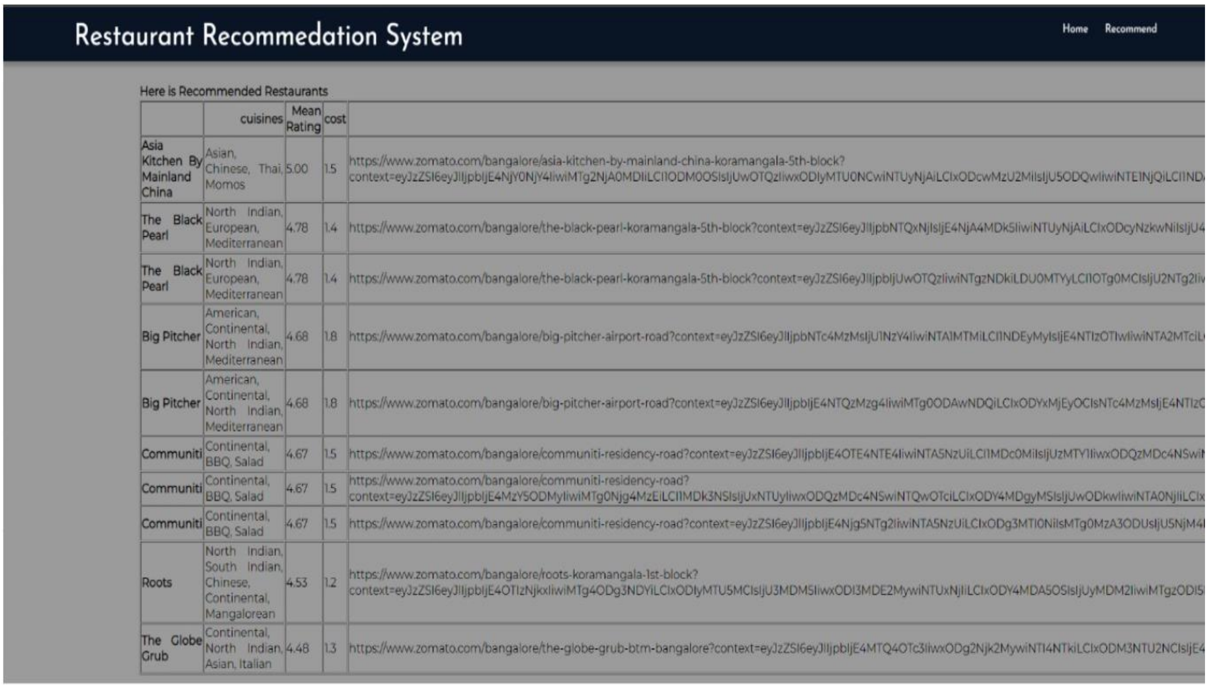
9. RESULTS

This is the home main page that describes the project and summarizes it.



Checking recommendation for the restaurant: 'Jalsa'

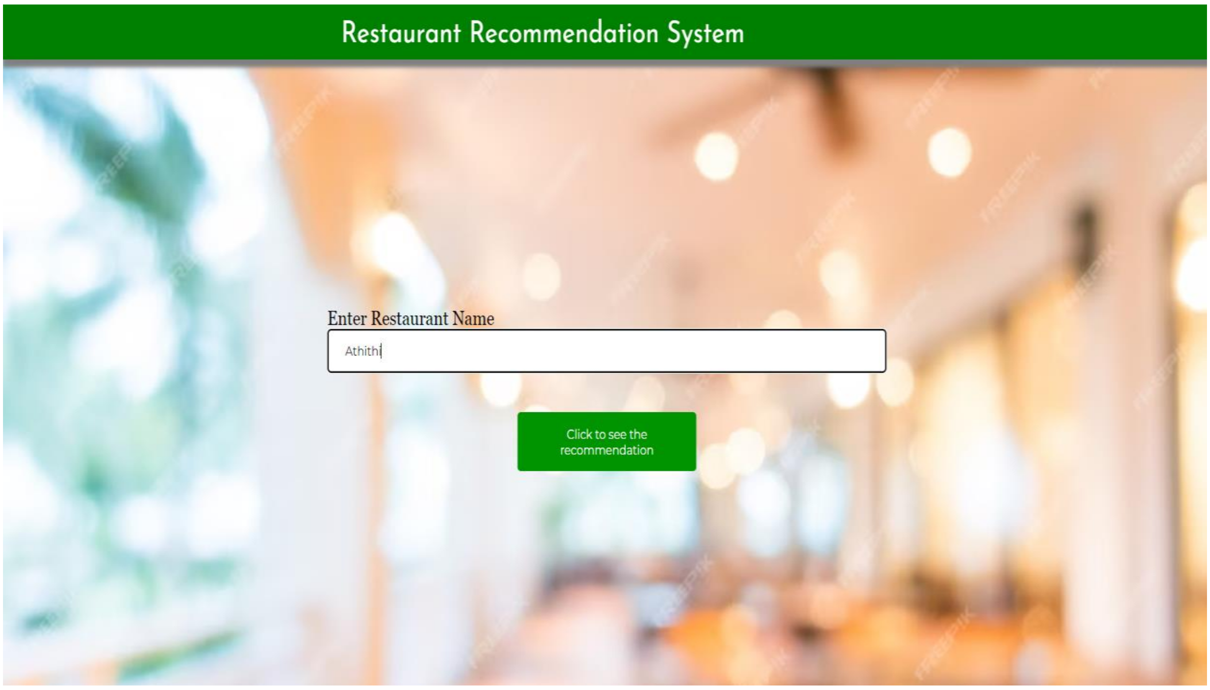


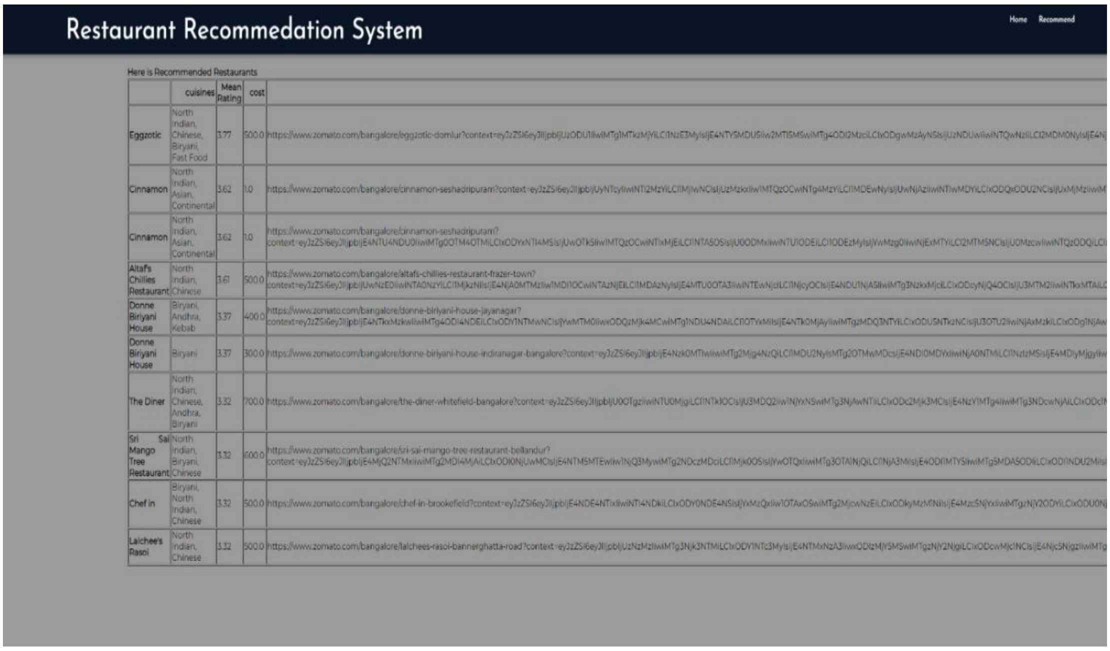


	cuisines	Mean Rating	cost	
Asia Kitchen By Mainland China	Asian, Chinese, Thai, Momos	5.00	1.5	https://www.zomato.com/bangalore/asia-kitchen-by-mainland-china-koramangala-5th-block?context=[illegible]
The Black Pearl	North Indian, European, Mediterranean	4.78	1.4	https://www.zomato.com/bangalore/the-black-pearl-koramangala-5th-block?context=[illegible]
The Black Pearl	North Indian, European, Mediterranean	4.78	1.4	https://www.zomato.com/bangalore/the-black-pearl-koramangala-5th-block?context=[illegible]
Big Pitcher	American, Continental, North Indian, Mediterranean	4.68	1.8	https://www.zomato.com/bangalore/big-pitcher-airport-road?context=[illegible]
Big Pitcher	American, Continental, North Indian, Mediterranean	4.68	1.8	https://www.zomato.com/bangalore/big-pitcher-airport-road?context=[illegible]
Communiti	Continental, BBQ, Salad	4.67	1.5	https://www.zomato.com/bangalore/communiti-residency-road?context=[illegible]
Communiti	Continental, BBQ, Salad	4.67	1.5	https://www.zomato.com/bangalore/communiti-residency-road?context=[illegible]
Communiti	Continental, BBQ, Salad	4.67	1.5	https://www.zomato.com/bangalore/communiti-residency-road?context=[illegible]
Roots	North Indian, South Indian, Chinese, Continental, Mangalorean	4.53	1.2	https://www.zomato.com/bangalore/roots-koramangala-1st-block?context=[illegible]
The Globe Grub	Continental, North Indian, Asian, Italian	4.48	1.3	https://www.zomato.com/bangalore/the-globe-grub-btm-bangalore?context=[illegible]

This is the page for predictions, where we input a restaurant's name to receive the top recommended restaurants. Recommendations are based on factors like cuisines, average rating (on a scale of 5), and cost in thousands.

Checking recommendation for the restaurant 'Athithi'





	cuisines	Mean Rating	cost	
Eggzotic	North Indian, Chinese, Biryani, Fast Food	3.77	500.0	https://www.zomato.com/bangalore/eggzotic-domlur?context=[illegible]
Cinnamon	North Indian, Asian, Continental	3.62	1.0	https://www.zomato.com/bangalore/cinnamon-seshadripuram?context=[illegible]
Cinnamon	North Indian, Asian, Continental	3.62	1.0	https://www.zomato.com/bangalore/cinnamon-seshadripuram?context=[illegible]
Altaf's Chillies Restaurant	North Indian, Chinese	3.61	500.0	https://www.zomato.com/bangalore/altafs-chillies-restaurant-frazer-town?context=[illegible]
Donne Biriyani House	Biryani, Andhra, Kebab	3.37	400.0	https://www.zomato.com/bangalore/donne-biriyani-house-jayanagar?context=[illegible]
Donne Biriyani House	Biryani	3.37	300.0	https://www.zomato.com/bangalore/donne-biriyani-house-indiranagar-bangalore?context=[illegible]
The Diner	North Indian, Chinese, Andhra, Biryani	3.32	700.0	https://www.zomato.com/bangalore/the-diner-whitefield-bangalore?context=[illegible]
Sri Sai Mango Tree Restaurant	North Indian, Biryani, Chinese	3.32	600.0	https://www.zomato.com/bangalore/sri-sai-mango-tree-restaurant-bellandur?context=[illegible]
Chef in	Biryani, North Indian, Chinese	3.32	500.0	https://www.zomato.com/bangalore/chef-in-brookefield?context=[illegible]
Laichee's Rasoi	North Indian, Chinese	3.32	500.0	https://www.zomato.com/bangalore/laichees-rasoi-bannerghatta-road?context=[illegible]

Finally, the prediction for the given restaurant inputs is shown.

10.ADVANTAGES AND DISADVANTAGES

Advantages:

• Experimenting with different cuisines is an option.

• Preparing meals is not mandatory.

• You have the opportunity to socialize with loved ones.

• It's simpler to cater to big gatherings.

• There's no squandering of time.

Disadvantages:

• Substantial financial commitments needed.

• An abundance of options available.

• A complicated initial setup procedure.

• Deficiency in data analytics capacity.

• The challenge of starting from scratch.

• Unable to track shifts in user conduct.

11.CONCLUSION

The primary aim of this research is to create an innovative restaurant recommendation system by leveraging machine learning integrated with a user-friendly web interface, essentially functioning as an application designed for customers. This application is intended to assist users in predicting suitable restaurants and identifying the most popular dishes based on geographical regions and individual preferences. It ensures that customers have access to restaurant ratings. By employing both popularity-based and collaborative-based filtering techniques, the recommendation system is optimized to enhance efficiency, enabling every user to effortlessly find suitable restaurants.

One common scenario often involves users seeking restaurants in close proximity to their current location. We're addressing this requirement by incorporating restaurant locations into our dataset. By doing so, our machine learning algorithm becomes adept at predicting suitable restaurants for customers based on their present location.

The envisioned restaurant recommendation system, implemented as a web application, is designed to significantly enhance the user experience when searching for restaurants. Its main focus is to provide efficient and rapid restaurant suggestions based on proximity, thereby reducing user effort and optimizing time management.

This innovative system not only forecasts appropriate restaurants but also offers insights into popular regional dishes, catering to individual tastes. By leveraging machine learning algorithms, the application ensures that users receive personalized and tailored recommendations. The use of popularity-based and collaborative-based filtering techniques serves to refine and optimize the accuracy of suggestions provided to users.

Moreover, the integration of location data within the dataset significantly improves the system's predictive capabilities. This allows for precise recommendations, considering a user's present location, thus saving time and effort that would otherwise be spent in manually searching for nearby restaurants.

The fundamental objective is to streamline the process of finding a restaurant by empowering users with a user-friendly and efficient tool. By simplifying the search for a dining establishment, users can spend less time browsing through various options and, instead, rely on the system's accurate and personalized suggestions. This expedites the decision-making process for the user, consequently making their time more valuable and saving them from unnecessary hassle.

In essence, this comprehensive system aims to revolutionize the way users search for and select restaurants by introducing an intelligent, user-centric approach. It seeks to diminish the complexities of decision-making in restaurant selection and contribute to an enhanced dining experience for users.

12.FUTURE SCOPE

The subsequent objective for an upcoming project is to enhance the system's performance and augment member benefits by introducing additional member functions. These could include features such as an online reservation system, an expanded online menu for ordering, and the development of a website, among other enhancements.

13. BIBILOGRAPHY

https://medium.com/mlearning-ai/restaurant-recommendation-

system-based-on-the-contentin-reviews-dfc3351004db

https://www.kaggle.com/code/midouazerty/restaurant-recommendation-system-using-ml

https://towardsdatascience.com/yelp-restaurant-recommendation-system-capstone-project-264fe7a7dea1

https://www.academia.edu/85069823/RESTAURANT_RECOMMENDATION_SYSTEM

14. APPENDIX

Source Code

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import plotly.offline as py
import plotly.graph_objs as go
import seaborn as sns
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
import nltk
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [3]:

```
zomato_data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/zomato.csv")
zomato_df=zomato_data.copy()
zomato_df.head(2)
```

Out[3]:

	url	address	name	online_order	book_table	rate	votes	phone	location
0	https://www.zomato.com/bangalore/jalsa-	942, 21st Main Road, 2nd Stage.	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91	Banashankari

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

Out[3]:

	url	address	name	online_order	book_table	rate	votes	phone	location
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari

In [4]:
```
zomato_df.shape
```

Out[4]:
```
(51717, 17)
```

In [5]:
```
zomato_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   url                          51717 non-null  object
 1   address                      51717 non-null  object
 2   name                         51717 non-null  object
 3   online_order                 51717 non-null  object
 4   book_table                   51717 non-null  object
 5   rate                         43942 non-null  object
 6   votes                        51717 non-null  int64
 7   phone                        50509 non-null  object
 8   location                     51696 non-null  object
 9   rest_type                    51490 non-null  object
 10  dish_liked                   23639 non-null  object
 11  cuisines                     51672 non-null  object
 12  approx_cost(for two people)  51371 non-null  object
 13  reviews_list                 51717 non-null  object
 14  menu_item                    51717 non-null  object
```

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

```
 14  menu_item                    51717 non-null  object
 15  listed_in(type)              51717 non-null  object
 16  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

In [6]:
```
zomato_df.isnull().sum()
```

Out[6]:
```
url                                0
address                            0
name                               0
online_order                       0
book_table                         0
rate                            7775
votes                              0
phone                           1208
location                          21
rest_type                        227
dish_liked                     28078
cuisines                          45
approx_cost(for two people)      346
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

In [7]:
```
#Dropping the column "dish_liked", "phone", "url"
zomato_df=zomato_df.drop(['phone','dish_liked'],axis=1)

#Remove the NaN values from the dataset
zomato_df.dropna(how='any',inplace=True)

#Removing the Duplicates
zomato_df.duplicated().sum()
zomato_df.drop_duplicates(inplace=True)

#Changing the column names
zomato_df = zomato_df.rename(columns={'approx_cost(for two people)':'cost','listed_in(type)':'type', 'listed_in(city)':'cit

#Removing '/5' from Rates
zomato_df = zomato_df.loc[zomato_df.rate !='NEW']
zomato_df = zomato_df.loc[zomato_df.rate !='-'].reset_index(drop=True)
remove_slash = lambda x: x.replace('/5', '') if type(x) == np.str else x
zomato_df.rate = zomato_df.rate.apply(remove_slash).str.strip().astype('float')
```

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

```python
#Changing the cost to string
zomato_df['cost'] = zomato_df['cost'].astype(str)
zomato_df['cost'] = zomato_df['cost'].apply(lambda x: x.replace(',','.'))
zomato_df['cost'] = zomato_df['cost'].astype(float)
```

In [8]:
```python
zomato_df.shape
```

Out[8]: (41263, 15)

In [9]:
```python
zomato_df.isnull().sum()
```

Out[9]:
```
url            0
address        0
name           0
online_order   0
book_table     0
rate           0
votes          0
location       0
rest_type      0
cuisines       0
cost           0
reviews_list   0
menu_item      0
type           0
city           0
dtype: int64
```

In [10]:
```python
## Computing Mean Rating
restaurants = list(zomato_df['name'].unique())
zomato_df['Mean Rating'] = 0
for i in range(len(restaurants)):
    zomato_df['Mean Rating'][zomato_df['name'] == restaurants[i]] = zomato_df['rate'][zomato_df['name'] == restaurants[i]].
#Scaling the mean rating values
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (1,5))
zomato_df[['Mean Rating']] = scaler.fit_transform(zomato_df[['Mean Rating']]).round(2)
```

In [11]:
```python
zomato_df[['name','rate','Mean Rating']].head()
```

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

Out[11]:

	name	rate	Mean Rating
0	Jalsa	4.1	3.99
1	Spice Elephant	4.1	3.97
2	San Churro Cafe	3.8	3.58
3	Addhuri Udupi Bhojana	3.7	3.45
4	Grand Village	3.8	3.58

In [12]:
```python
## Lower Casing
zomato_df["reviews_list"] = zomato_df["reviews_list"].str.lower()

## Removal of Puctuations
import string
PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))
zomato_df["reviews_list"] = zomato_df["reviews_list"].apply(lambda text: remove_punctuation(text))
```

In [13]:
```python
zomato_df[['reviews_list', 'cuisines','url']].sample(5)
```

Out[13]:

	reviews_list	cuisines	url
2565	rated 20 ratedn bad quality of puff and bomba...	Bakery	https://www.zomato.com/bangalore/cake-art-basa...
32714	rated 40 ratedn limited options in the menuth...	Continental, North Indian, Chinese, Arabian	https://www.zomato.com/bangalore/high-sky-whit...
12842	rated 30 ratedn rude behavior by the staff ve...	North Indian, Mughlai, Mediterranean, Iranian	https://www.zomato.com/bangalore/ruh-bellandur...
30607	rated 30 ratedn been there on several occasio...	North Indian, Mithai	https://www.zomato.com/bangalore/bhaiyaji-food...
29166	rated 10 ratedn service was very disappointin...	Chinese, Thai, Asian	https://www.zomato.com/bangalore/magnolia-kora...

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

In [14]:
```
def get_top_words(column, top_nu_of_words, nu_of_word):

    vec = CountVectorizer(ngram_range= nu_of_word, stop_words='english')

    bag_of_words = vec.fit_transform(column)

    sum_words = bag_of_words.sum(axis=0)

    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]

    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)

    return words_freq[:top_nu_of_words]
```

In [15]:
```
# RESTAURANT NAMES:
restaurant_names = list(zomato_df['name'].unique())
def get_top_words(column, top_nu_of_words, nu_of_word):
    vec = CountVectorizer(ngram_range= nu_of_word, stop_words='english')
    bag_of_words = vec.fit_transform(column)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:top_nu_of_words]

zomato_df=zomato_df.drop(['address','rest_type', 'type', 'menu_item', 'votes'],axis=1)


# Randomly sample 60% of your dataframe
df_percent = zomato_df.sample(frac=0.5)
```

In [16]:
```
zomato_df.head()
```

Out[16]:

	url	name	online_order	book_table	rate	location	cuisines	cost	reviews_list	
0	https://www.zomato.com/bangalore/jalsa-banasha...	Jalsa	Yes	Yes	4.1	Banashankari	North Indian, Mughlai, Chinese	800.0	rated 40 ratedn a beautiful place to dine int...	Ban

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

Out[16]:

	url	name	online_order	book_table	rate	location	cuisines	cost	reviews_list	
0	https://www.zomato.com/bangalore/jalsa-banasha...	Jalsa	Yes	Yes	4.1	Banashankari	North Indian, Mughlai, Chinese	800.0	rated 40 ratedn a beautiful place to dine int...	Ban
1	https://www.zomato.com/bangalore/spice-elephan...	Spice Elephant	Yes	No	4.1	Banashankari	Chinese, North Indian, Thai	800.0	rated 40 ratedn had been here for dinner with...	Ban
2	https://www.zomato.com/SanchurroBangalore?cont...	San Churro Cafe	Yes	No	3.8	Banashankari	Cafe, Mexican, Italian	800.0	rated 30 ratedn ambience is not that good eno...	Ban
3	https://www.zomato.com/bangalore/addhuri-udupi...	Addhuri Udupi Bhojana	No	No	3.7	Banashankari	South Indian, North Indian	300.0	rated 40 ratedn great food and proper karnata...	Ban
4	https://www.zomato.com/bangalore/grand-village...	Grand Village	No	No	3.8	Basavanagudi	North Indian, Rajasthani	600.0	rated 40 ratedn very good restaurant in neigh...	Ban

In [17]:
```
zomato_df.to_csv("restaurant1.csv")
```

In [18]:
```
zomato_df.to_csv("restaurant2.csv")
```

In [19]:
```
df_percent.head()
```

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

Out[19]:

	url	name	online_order	book_table	rate	location	cuisines	cost	reviews_list
5676	https://www.zomato.com/bangalore/kfc-3-whitefi...	KFC	Yes	No	2.8	ITPL Main Road, Whitefield	Burger, Fast Food	400.0	rated 40 ratedn in banglore there are many kf...
31391	https://www.zomato.com/bangalore/corner-house-...	Corner House Ice Cream	Yes	No	4.4	Seshadripuram	Ice Cream, Desserts	400.0	rated 40 ratedn very close to my work place w...
5181	https://www.zomato.com/bangalore/just-bake-sha...	Just Bake	Yes	No	3.5	Shanti Nagar	Bakery, Desserts	400.0	rated 50 ratedn just bake cake is just awesom...
31359	https://www.zomato.com/bangalore/tea-samakruth...	Tea Samskruthi	No	No	3.9	Malleshwaram	Cafe	200.0	rated 50 ratedn tea samskruti is the best pla...
11609	https://www.zomato.com/bangalore/taco-bell-ind...	Taco Bell	Yes	No	4.1	Indiranagar	Mexican, American, Fast Food	600.0	rated 40 ratedn good place for mexican food t...

In [20]:

```
df_percent['reviews_list'].isnull().sum()
```

Out[20]: 0

In [21]:

```
df_percent['url'].isnull().sum()
```

Out[21]: 0

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

In [22]:

```
df_percent.set_index('name', inplace=True)
indices = pd.Series(df_percent.index)

# Creating tf-idf matrix
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'])

cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [23]:

```
def recommend(name, cosine_similarities = cosine_similarities):

    # Create a list to put top restaurants
    recommend_restaurant = []

    # Find the index of the hotel entered
    idx = indices[indices == name].index[0]

    # Find the restaurants with a similar cosine-sim value and order them from bigges number
    score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False)

    # Extract top 30 restaurant indexes with a similar cosine-sim value
    top30_indexes = list(score_series.iloc[0:31].index)

    # Names of the top 30 restaurants
    for each in top30_indexes:
        recommend_restaurant.append(list(df_percent.index)[each])

    # Creating the new data set to show similar restaurants
    df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost','url'])

    # Create the top 30 similar restaurants with some of their columns
    for each in recommend_restaurant:
        df_new = df_new.append(pd.DataFrame(df_percent[['cuisines','Mean Rating', 'cost','url']][df_percent.index == each].

    # Drop the same named restaurants and sort only the top 10 by the highest rating
    df_new = df_new.drop_duplicates(subset=['cuisines','Mean Rating', 'cost','url'], keep=False)
    df_new = df_new.sort_values(by='Mean Rating', ascending=False).head(10)

    print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' % (str(len(df_new)), name))

    return df_new
```

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

In [24]:
```
recommend('Pai Vihar')
```

TOP 10 RESTAURANTS LIKE Pai Vihar WITH SIMILAR REVIEWS:

Out[24]:

	cuisines	Mean Rating	cost	url
Cinnamon	North Indian, Asian, Continental	3.62	1.0	https://www.zomato.com/bangalore/cinnamon-sesh...
Samosa Singh	Street Food, Fast Food, Rolls, Desserts	3.60	200.0	https://www.zomato.com/bangalore/samosa-singh-...
Samosa Singh	Street Food, Beverages	3.60	150.0	https://www.zomato.com/bangalore/samosa-singh-...
Kadai Crust - Amma Veetu Samayal	Chettinad, South Indian, Biryani	3.58	700.0	https://www.zomato.com/bangalore/kadai-crust-a...
Pallavi Restaurant	Biryani, Chinese, Andhra	3.58	500.0	https://www.zomato.com/bangalore/pallavi-resta...
Upahar Sagar	South Indian, Chinese, North Indian	3.58	350.0	https://www.zomato.com/bangalore/upahar-sagar-...
Magix's Paratha Roll	Fast Food, North Indian, Chinese, Mughlai, Rolls	3.52	400.0	https://www.zomato.com/bangalore/magixs-paratt...
Magix's Paratha Roll	Fast Food, North Indian, Chinese, Mughlai, Rolls	3.52	400.0	https://www.zomato.com/bangalore/magixs-paratt...
Magix's Paratha Roll	Fast Food, North Indian, Chinese, Mughlai, Rolls	3.52	400.0	https://www.zomato.com/bangalore/magixs-paratt...
Prasiddhi Food Corner	Fast Food, North Indian, South Indian	3.45	200.0	https://www.zomato.com/bangalore/prasiddhi-foo...

In [25]:
```
recommend('Canopy')
```

TOP 10 RESTAURANTS LIKE Canopy WITH SIMILAR REVIEWS:

Out[25]:

	cuisines	Mean Rating	cost	url
Atithi	North Indian, Chinese, Street Food	3.63	800.0	https://www.zomato.com/bangalore/atithi-hsr?co...
Atithi	North Indian, Chinese, Street Food	3.63	800.0	https://www.zomato.com/bangalore/atithi-hsr?co...
Cinnamon	North Indian, Chinese, Biryani	3.62	550.0	https://www.zomato.com/bangalore/cinnamon-hsr?...

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

Out[25]:

	cuisines	Mean Rating	cost	url
Atithi	North Indian, Chinese, Street Food	3.63	800.0	https://www.zomato.com/bangalore/atithi-hsr?co...
Atithi	North Indian, Chinese, Street Food	3.63	800.0	https://www.zomato.com/bangalore/atithi-hsr?co...
Cinnamon	North Indian, Chinese, Biryani	3.62	550.0	https://www.zomato.com/bangalore/cinnamon-hsr?...
Cafe @ Elanza	Chinese, North Indian, Cafe	3.45	1.0	https://www.zomato.com/bangalore/cafe-elanza-r...
Cafe @ Elanza	Chinese, North Indian, Cafe	3.45	1.0	https://www.zomato.com/bangalore/cafe-elanza-r...
Nouvelle Garden	North Indian, Continental, Italian	3.45	900.0	https://www.zomato.com/bangalore/nouvelle-gard...
Sri Sai Mango Tree Restaurant	North Indian, Biryani, Chinese	3.32	600.0	https://www.zomato.com/bangalore/sri-sai-mango...
The Onyx - The HHI Select Bengaluru	North Indian, Chinese, Continental	2.97	950.0	https://www.zomato.com/bangalore/the-onyx-the-...
Wazir's	North Indian, Chinese	2.94	500.0	https://www.zomato.com/bangalore/wazirs-shanti...
Melange - Hotel Ekaa	North Indian, Chinese, Continental, Mangalorean	2.81	900.0	https://www.zomato.com/bangalore/melange-hotel...

In [26]:

```
recommend('Cinnamon')
```

```
TOP 10 RESTAURANTS LIKE Cinnamon WITH SIMILAR REVIEWS:
```

Out[26]:

Chinita Real

Preview Code Blame 1 lines (1 loc) · 111 KB Code 55% faster with GitHub Copilot Raw

Out[26]:

Chinita Real
Soda Bottl
CafÃƒÂƒÃ,ÂƒÃƒÂ,Ã,ÂƒÃƒÂƒÃ,Â,ÃƒÂ,Ã,ÂƒÃƒÂƒÃ,ÂƒÃƒÂ,Ã,Â,ÃƒÂƒÃ,Â,Ã
CafÃƒÂƒÃ,ÂƒÃƒÂ,Ã,ÂƒÃƒÂƒÃ,Â,ÃƒÂ,Ã,ÂƒÃƒÂƒÃ,ÂƒÃƒÂ,Ã,Â,ÃƒÂƒÃ,Â,Ã
CafÃƒÂƒÃ,ÂƒÃƒÂ,Ã,ÂƒÃƒÂƒÃ,ÃƒÂ,Ã,ÂƒÃƒÂƒÃ,ÂƒÃƒÂ,Ã,Â,ÃƒÂƒÃ,Â,ÃƒÂ,Ã,ÂƒÃƒÂƒÃ,ÂƒÃƒÂ,Ã,ÃƒÂƒÃ,Â,ÃƒÂ,Ã,Â,ÃƒÂƒÃ,ÂƒÃƒÂ,Ã,Â,ÃƒÂƒ
Foxtrot - House

In [27]:

```
import pickle
pickle.dump(tfidf, open('restaurant2.pkl', 'wb'))
```

In [ ]:

app.py

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import plotly.offline as py
import plotly.graph_objs as go
import seaborn as sns
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
import nltk
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import flask
from flask import Flask,render_template, request
import pickle

app = Flask(__name__) # initializing a flask app
model=pickle.load(open("restaurant2.pkl",'rb'))  #loading the model
```

```
#loading the updated dataset
zomato_df=pd.read_csv("restaurant2.csv")

@app.route('/')# route to display the home page
def home():
    return render_template('home.html')#rendering the home page

@app.route('/extractor')
def extractor():
    return render_template('extractor.html')

#extractor page

@app.route('/keywords',  methods=['POST'])
def keywords():
    output = request.form['output']

    print(output)
    print(type(output))

    df_percent = zomato_df.sample(frac=0.5)
    df_percent.set_index('name', inplace=True)
    indices = pd.Series(df_percent.index)
```

```
    tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2),
min_df=1, stop_words='english')  # Change min_df to 1
    tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'].fillna('
'))
    cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

    def recommend(name, cosine_similarities=cosine_similarities):
        recommend_restaurant = []
        idx = indices[indices == name].index[0]
        score_series =
pd.Series(cosine_similarities[idx]).sort_values(ascending=False)
        top30_indexes = list(score_series.iloc[0:31].index)

        for each in top30_indexes:
            recommend_restaurant.append(list(df_percent.index)[each])

        df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating',
'cost','url'])

        for each in recommend_restaurant:
            df_new = pd.concat([df_new, df_percent[['cuisines','Mean
Rating', 'cost','url']][df_percent.index == each].sample()])
```

```
        df_new = df_new.drop_duplicates(subset=['cuisines','Mean
Rating', 'cost','url'], keep=False)
        pd.set_option('display.max_columns', None)


        df_new = df_new.sort_values(by='Mean Rating',
ascending=False).head(10)  # Fix the sorting line
        print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: '
% (str(len(df_new)), name))


        return df_new


    result = recommend(output)
    print(result)
    print(type(result))


    return render_template('keywords.html',
keyword=result.to_html())


if __name__ == "__main__":
   # running the app
    app.run(debug=True)
```

Github:-

https://github.com/smartinternz02/SI-GuidedProject-594152-1697280123/tree/main

Project Demo Link:-

https://drive.google.com/file/d/1lIr4S0CcQOz3l0w0OBiWRGmnKGAELJ5N/view?usp=sharing