RESTAURANT RECOMMENDATION SYSTEM

Project Description:-

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.

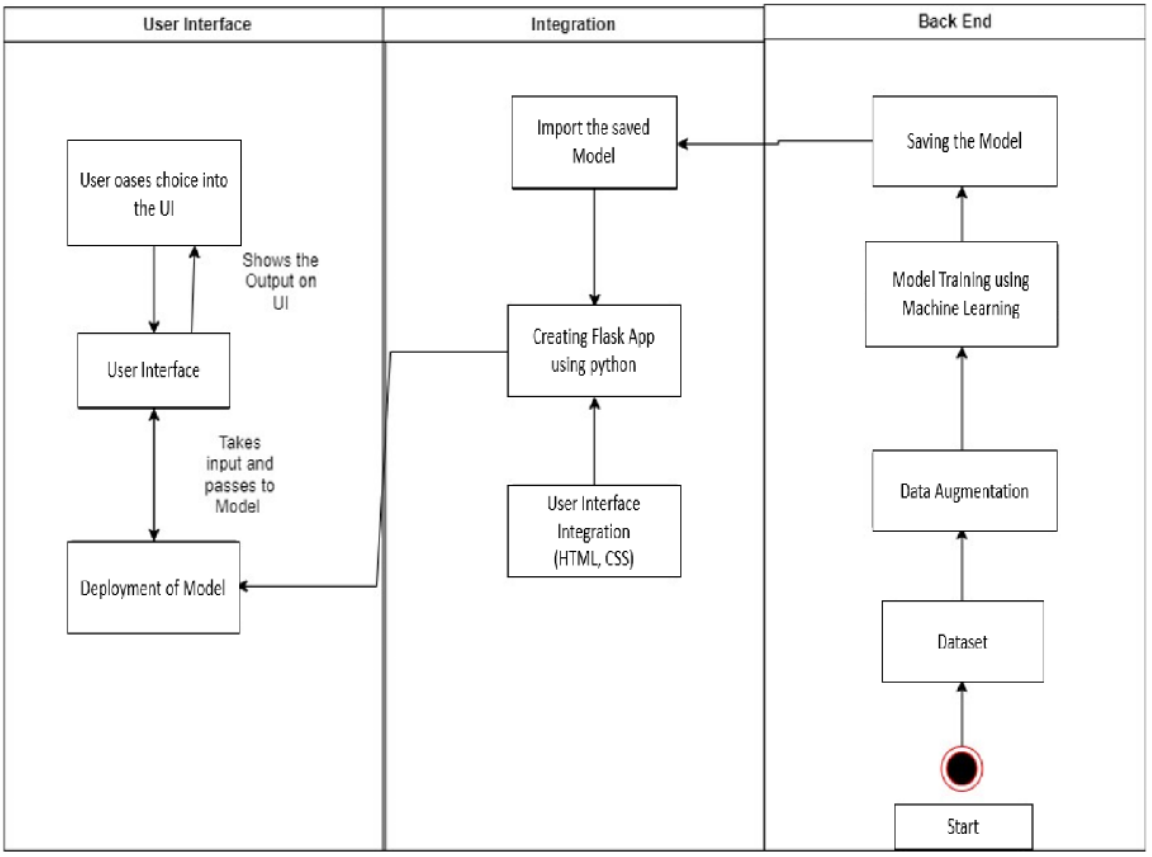
Solution:-

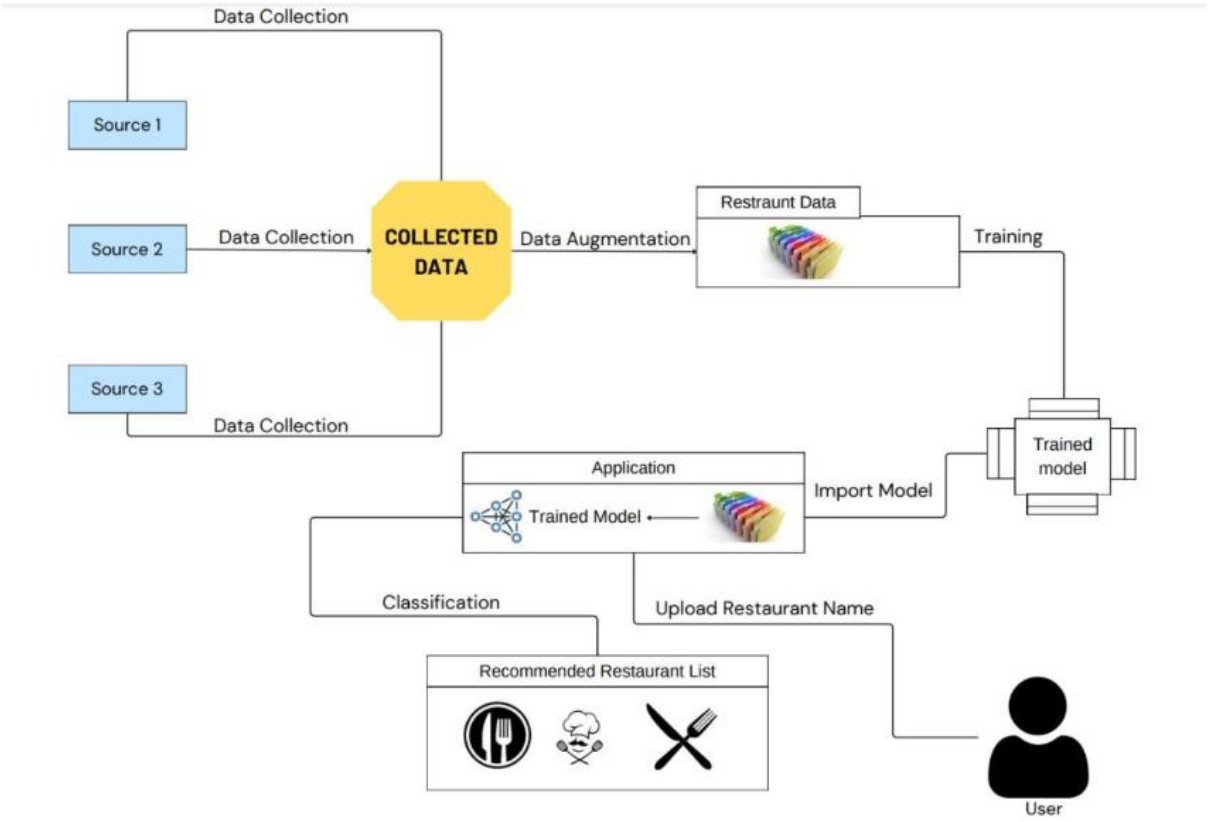
Here, we're developing a recommendation system that is content-based. The goal is to develop a content-based recommender system where, when we enter the name of a restaurant, the system looks at the reviews of related restaurants and suggests additional restaurants to us, sorting them by highest rating. Travelers who are unfamiliar with a city will mostly gain from this recommendation system. During their visit to a city, the majority of tourists always enjoy dining at renowned restaurants.

If not, residents in the same city might utilize it extensively to check if any new restaurants are suggested based on their behaviour.

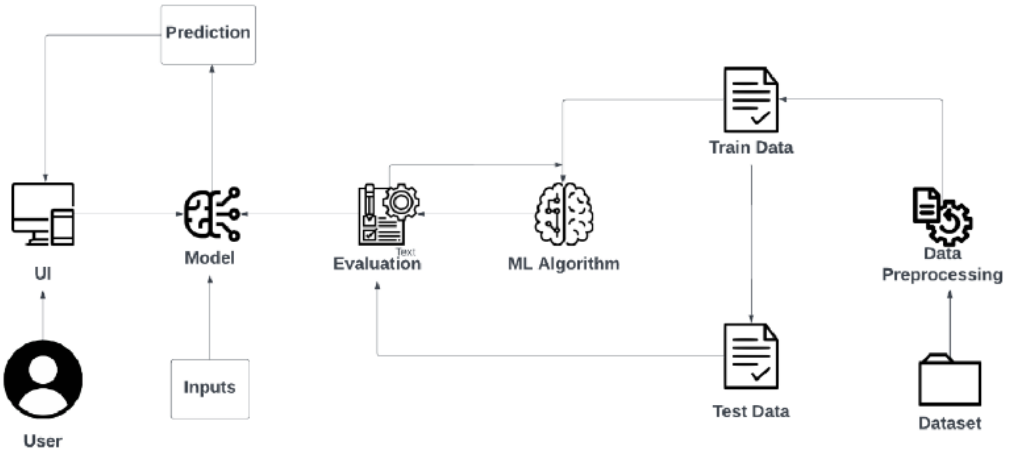
Architecture:-

Technical Architecture:





Solution Architecture Diagram:-



LEARNING OUTCOMES

After completing this project, you will be able to:

- Execute one of the methods to create your own recommendation system.
- You will be able to learn about the Content-Based Filtering recommendation method.
- Using various data pre-processing approaches, you will be able to know how to pre-process / clean the data.
- Data visualization will enable you to analyze or gain insights from the data.
- Algorithms are applied based on visualization and datasets.
- You will be able to determine the model's correctness.
- You'll learn how to use the Flask framework to create a web application.

Pre-Requisites:-

To complete the project successfully, you need to install following software & packages:

Activity 1:- Install VS Code/Use Google Collab

- In order to develop a solution to this problem and to run and test it we need the appropriate environment to do this.
- We use Visual Studio Code to do this. Or you can also use Google Collab to pre-process your data and train the model then use it in VS code by using flask

Activity 2:- To build the machine learning model you need the following packages

- **Numpy:-**

  It's a Python machine learning package that's free. It includes a number of different algorithms, including random forests, k-neighbors, and support for Python scientific and numerical libraries, such as NumPy and SciPy.

- **Matplotlib and seaborn:-**

  The primary use of Matplotlib is for simple plotting. When using Matplotlib, visualizations typically include scatter plots, bars, pies, lines, and other shapes. Seaborn: In contrast, Seaborn offers a range of visualization patterns. Its default themes are readily engaging and it employs less syntax overall.

- **Flask:-**

  It is a web framework used for building web application we will use it with our train model files to create our web application for our restaurant recommendation system.

If you are using visual studio, follow below steps to download the required packages:

- Open VS code terminal
- Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install seaborn” and click enter.
- Type “pip install plotly” and click enter.
- Type “pip install numpy” and click enter.
- Type “pip install scikit-image” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install Flask” and click enter.

Prior Knowledge:-

One should know about the following concepts before starting the project:

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

Project Work Flow:-

● The user enters input features by interacting with the User Interface (UI).

● The model that is incorporated analyzes entered features or input.

● The prediction is displayed on the user interface (UI) when the model has processed the inputs.

Image

Task :-

1: Data Collection.

● Compile or generate the dataset

2. Pre-processing of data.

- Bring the Libraries in.
- bringing in the dataset.
- Investigative Data Analysis
- Visualization of Data.

3. Filtering by Content

- combining datasets
- Building the recommendation engine
- Forecasting the outcome

4. Developing Applications

- Make a file in HTML.
- Construct a Python Code

First Milestone: Gathering Data

The 1st milestone now focuses on the generation or gathering of datasets.

Zomato Bangalore will be the source of our analysis and the content filtering approach will be used to make results.

The dataset link is as follows:
https://www.kaggle.com/datasets/himanshupoddar/zomato-bangalore-restaurants

Milestone 2: Data Pre-processing

In this milestone, you need to complete all the below activities to build the model:-

First Task: Bringing in Libraries

To create a recommendation system and preprocess data, import the libraries listed below. NumPy and Pandas are used for pre-processing and cleaning data. Bar plots and other visual representations for the dataset were made possible via Seaborn, Plotly, and Matplotlib. Additionally, as text data (reviews) would also need to be cleaned, the nltk and sklearn libraries will be used for that purpose.

```
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
```

Activity 2: Read the Dataset:

As we are using google collab to pre-process and train our model we will need to put the dataset in our google drive and then mount our drive in it from there we will be able to read our dataset.

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

```
Mounted at /content/drive
```

The files in our dataset may be in the.csv, excel,.txt, json, or other formats. Pandas can assist us in reading the dataset.

To read the dataset, use the read_csv() function in pandas. We must provide the directory for the CSV file as a parameter.

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

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

	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	reviews_list	menu_item	listed_in(type)	listed_in(city)
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	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Buffet	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	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800	[('Rated 4.0', 'RATED\n Had been here for din...	[]	Buffet	Banashankari

Activity 3: Analyse the dataset:

Checking dataset size

```
zomato_df.shape
```

[4]

```
(51717, 17)
```

The dataset has 51717 records with 17 features

Checking the columns in the dataset:

```
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
 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
```

Description for columns

1. URL is the restaurant's URL on the Zomato platform.

2. address includes the Bengaluru restaurant's address

3. name includes the restaurant's name.

4. online_order Whether the establishment offers online ordering or not

5. Is the book_table table book option accessible?

6. rate is the restaurant's overall rating out of 5.

7.Vote contains the total number of rating for the restaurant as of the above mentioned date

8. phone has the restaurant's phone number on it.

9. location includes the neighborhood where the eatery is situated.

10. rest_type:- restaurant type

11. dish_liked :-food dishes patrons enjoyed at the eatery

12. cuisines:- food type

13. The approximate cost of a dinner for two people is contained in the approx_cost(for two persons) variable.

14. reviews_list is a list of tuples with restaurant reviews in each tuple.

15. A list of the menus the restaurant offers is contained in menu_item.

16. listed_in(type) type of meal

17. The neighborhood where the restaurant is listed is contained in listed_in(city).

Understanding Overview of features

➔ **DataFrame Structure and Data Types:**

(i) The structure of data in a DataFrame or Python object significantly influences data handling and processing.
(ii) Two primary data types are prevalent: numeric and textual.
(iii) Elaboration: The way data is organized within a DataFrame or Python object affects how various operations and analyses can be conducted on that data. Numeric data types include integers (whole numbers) and floats (numbers with decimals), while textual data, referred to as Strings in Python or Objects in Pandas, encompass characters and numbers, representing words, sentences, or larger text structures.

➔ **Characteristics of Numeric and Textual Data:**

(i) Numeric data comprises integers and floats, representing numerical values.
(ii) Textual data, specifically Strings in Python or Objects in Pandas, encompass text-based information that can contain both characters and numbers.
(iii) Elaboration: Numeric data refers to numerical values that can be used for mathematical operations, whereas textual data represents information in the form of text, allowing a broader

scope of data representation, including words, sentences, or even more complex textual structures.

➔ **Utilizing the info() Method:**

(i) The info() method in Python helps in understanding the structure and details of the dataset.
(ii) Elaboration: The info() method provides a summary of the dataset, offering insight into the types of data present in each column, their counts, and whether there are missing values. It assists in understanding the dataset's composition, aiding in the initial assessment and processing of data.

Categorical Data Consideration:

(i) In the dataset, most features, except 'votes', are categorized as categorical data.
(ii) Some seemingly continuous data might be represented as numerical but could actually be categorical.
(iii) Elaboration: While many features are considered categorical, some data might appear as continuous but could, in fact, be discrete or categorical in nature. These categorical values might be represented as numbers, which can lead to confusion or misinterpretation during analysis if not identified and handled properly.

In essence, comprehending the nature of data, its types, and how it's stored within a DataFrame or Python object is fundamental for conducting accurate analyses and handling diverse data structures effectively. The info() method provides a valuable initial overview of the dataset, but special attention is necessary when dealing with seemingly continuous data that might actually be categorical in nature.

```
zomato_df.info()
```

```
[5]
```

Python

```
<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
 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
```

Checking for null values in the dataset

```
zomato_df.isnull().sum()
```

```
[6]
```

Python

```
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
```

Data cleaning as our dataset contains null values and some special characters

```
#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)':'city'})

#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')

#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)
```

```
[7]
```

Python

Checking for null values after cleaning & data Processing

```
zomato_df.isnull().sum()
```

```
[6]
Python
```

```
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
```

Checking mean rating with restaurant name and rating for each restaurant using below line codes

```
## 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]].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)
```

We will be using the 'Review' and 'Cuisines' feature in order to create a recommender system. So we need to prepare and clean the text in those columns.

Operations performed: Lower Casing, Removal of Punctuations, Removal of Stop words, Removal of URLs, Spelling correction

```
## 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))
```

```
[12]
Python
```

```
zomato_df[['reviews_list', 'cuisines','url']].sample(5)
```

```
[13]
Python
```

	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...

Milestone 3: Data Visualization

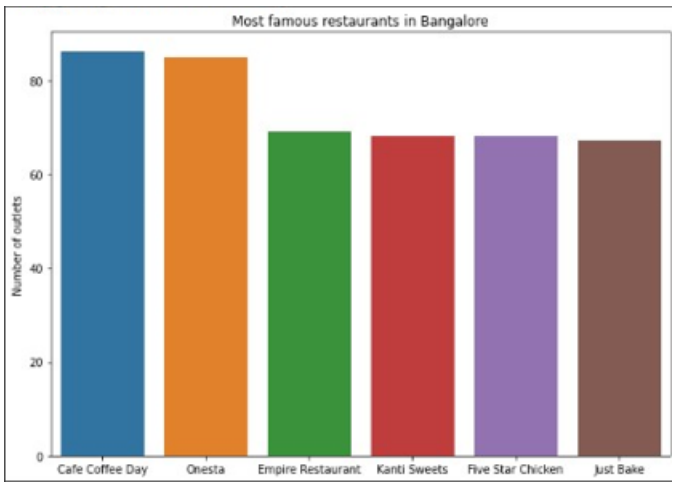
Data visualization involves presenting a given dataset in graphical form, aiding in identifying patterns, trends, and correlations that might be overlooked when dealing

solely with text-based data. Understanding the data and its internal relationships is equally crucial as the algorithms utilized for training in machine learning. Even the most advanced machine learning models can underperform if the data hasn't been properly visualized and comprehended.

To visualize the dataset, specific libraries such as Matplotlib and Seaborn are required. Matplotlib, a Python 2D plotting library, enables the creation of various graphical representations like plots, scatter plots, histograms, and bar charts.

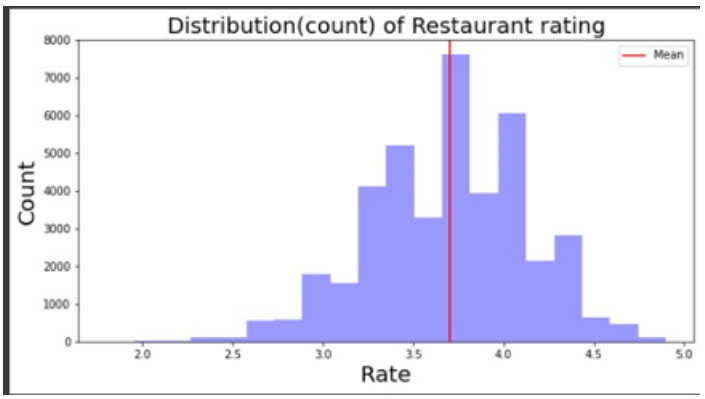
In order to implement visualization, we'll be using Matplotlib and the Seaborn library. Initially, we'll create a bar plot using Matplotlib to display the top six restaurants in Bangalore based on their value counts.

```
#Most Famous 6 restaurants in Banglore

plt.figure(figsize=(10,7))

chains=zomato_df['name'].value_counts()[:6]

sns.barplot(x=chains.index,y=chains,palette='tab10')

plt.title("Most famous restaurants in Bangalore")

plt.ylabel("Number of outlets")
```



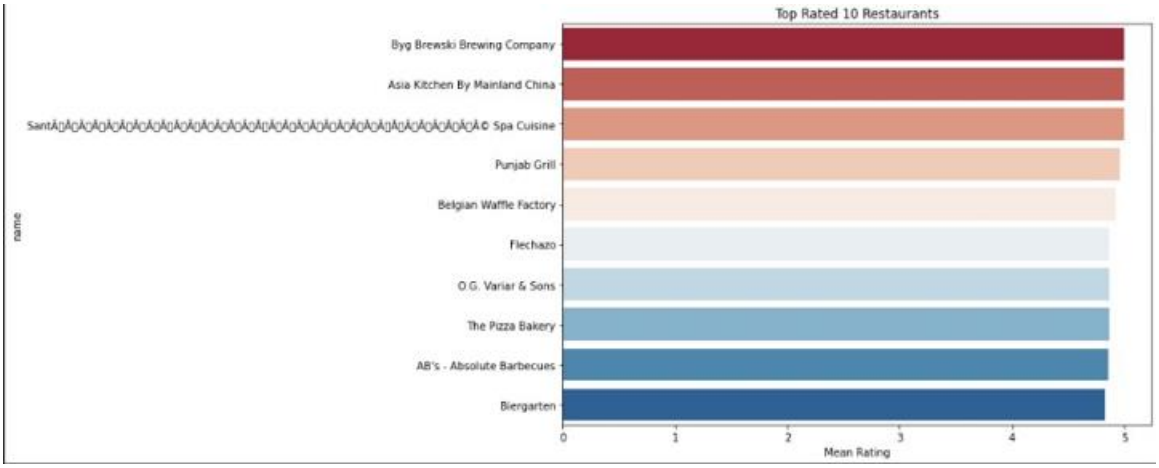
Checking the distribution of restaurant rating, for that we are using distplot from seaborn library.

```
#Distribution of Restaurant Rating

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10, 5))
sns.distplot(zomato_df.rate,kde=False,color = 'b',ax =ax,bins=20);
ax.axvline(zomato_df.rate.mean(), 0, 1, color='r', label='Mean')
ax.legend();
ax.set_ylabel('Count',size=20)
ax.set_xlabel('Rate',size=20)
ax.set_title('Distribution(count) of Restaurant rating',size=20);
```



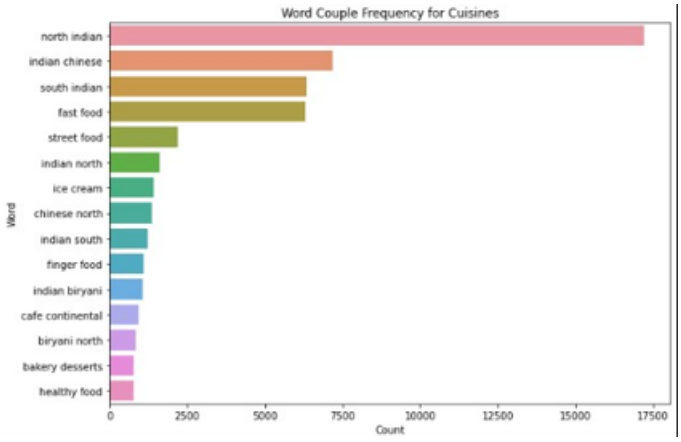
And we can infer that most of the restaurants in Bangalore have rating above 3.5. Visualizing top 10 rated restaurants in Bangalore. For that we are again using barplot from Matplotlib library.

```
# Top 10 Rated Restaurants

df_rating = zomato_df.drop_duplicates(subset='name')
df_rating = df_rating.sort_values(by='Mean Rating', ascending=False).head(10)
plt.figure(figsize=(10,7))
sns.barplot(data=df_rating, x='Mean Rating', y='name', palette='RdBu')
plt.title('Top Rated 10 Restaurants');
```



Visualizing two word frequencies for cuisines, using barplot from seaborn library.

```
[ ] # Top 15 two word frequencies for Cuisines
    lst = get_top_words(zomato_df['cuisines'], 15, (2,2))
    df_words = pd.DataFrame(lst, columns=['Word', 'Count'])
    plt.figure(figsize=(10,7))
    sns.barplot(data=df_words, x='Count', y='Word')
    plt.title('Word Couple Frequency for Cuisines');
```



Here we can see the Top favourite cuisine among people of Bangalore is 'North Indian', 'Indian Chinese' and 'Fast food'.

Milestone 4: CONTENT-BASE RECOMMENDER SYSTEM

Activity:1 TF-IDF Matrix (Term Frequency — Inverse Document Frequency Matrix)

TF-IDF is a statistical method of assessing the meaning of a word in a given document. Now we use TF-IDF vectorization on the dataset.

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. Calculating the cosine similarity of each item with every other item in the dataset.

```
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)
```

[22] Python

Here, the `tf-idf matrix` is the matrix containing each word and its TF-IDF score with regard to each document, or item in this case. Also, stop words are simply words that add no significant value to our system, like 'an', 'is', 'the', and hence are ignored by the system.

Calculating Cosine Similarity

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The formula for Cosine Similarity

And in the last line of code, we are calculating the cosine similarity of each item with every other item in the dataset. So we just pass the matrix as an argument.

Activity :2 Creating Recommendation system

```
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].sample()))

    # 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
```

[23] Python

Querying recommendation for 3 Restaurants:

For Restaurant ‘Pai Vihar’

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 Parattha Roll	Fast Food, North Indian, Chinese, Mughlai, Rolls	3.52	400.0	https://www.zomato.com/bangalore/magixs-paratt...
Magix's Parattha Roll	Fast Food, North Indian, Chinese, Mughlai, Rolls	3.52	400.0	https://www.zomato.com/bangalore/magixs-paratt...
Magix's Parattha 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...

For Restaurant 'Canopy'

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?...
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...

For Restaurant 'Cinnamon'

In [26]:
```
recommend('Cinnamon')
```

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

Out[26]:

	cuisines	Mean Rating	cost	url
Chianti	Italian	4.59	1.5	https://www.zomato.com/bangalore/chianti-koram...
Chianti	Italian	4.59	1.5	https://www.zomato.com/bangalore/chianti-mg-ro...
Chinita Real Mexican Food	Mexican	4.47	1.2	https://www.zomato.com/bangalore/chinita-real-...
Oh! Calcutta	Bengali, Seafood	4.39	1.2	https://www.zomato.com/bangalore/oh-calcutta-c...
Oh! Calcutta	Bengali, Seafood	4.39	1.2	https://www.zomato.com/bangalore/oh-calcutta-c...
Soda Bottle Opener Wala	Parsi, North Indian	4.36	1.3	https://www.zomato.com/bangalore/soda-bottle-o...
Ã‚ÃƒÂƒÃ‚Â‚Ã‚ÃƒÂƒÃ‚Â‚ÃƒÂƒÃ‚Â‚Ã‚Â‚ÃƒÂƒÃ‚Â‚Ã‚ÃƒÂ‚Ã‚Â‚Ã‚Â© Felix	American, Cafe, Continental	4.35	1.7	https://www.zomato.com/bangalore/caf%C3%A9-fel...
Ã‚ÃƒÂƒÃ‚Â‚Ã‚ÃƒÂƒÃ‚Â‚ÃƒÂƒÃ‚Â‚Ã‚Â‚ÃƒÂƒÃ‚Â‚Ã‚ÃƒÂ‚Ã‚Â‚Ã‚Â© Felix	American, Cafe, Continental	4.35	1.7	https://www.zomato.com/bangalore/caf%C3%A9-fel...
rÃƒÂ‚Ã‚ÃƒÂƒÃ‚Â‚Ã‚ÃƒÂƒÃ‚Â‚Ã‚Â‚ÃƒÂƒÃ‚Â‚ÃƒÂƒÃ‚Â‚Ã‚Â‚ÃƒÂƒÃ‚Â‚Ã‚ÃƒÂ‚Ã‚Â‚Ã‚Â© Felix	American, Cafe, Continental	4.35	1.7	https://www.zomato.com/bangalore/caf%C3%A9-fel...
Foxtrot - House of Subculture	Cafe, American, Asian, North Indian	4.35	1.0	https://www.zomato.com/bangalore/foxtrot-house...

Milestone 5: Application Building

Activity 1: Create an HTML File

We use HTML to create the front end part of the web page.

Here, we created 3 html pages - home.html, extractor.html and keywords.html. home.html displays home page, extractor.html accepts the values from the input and kewords.html displays the prediction.

Home.html code:

```
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Restaurant Recommendation System</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
<style>
.header {
            top:0;
            margin:0px;
            left: 0px;
            right: 0px;
            position: fixed;
            background-color: #091425;
            color: white;
            box-shadow: 0px 8px 4px grey;
            overflow: hidden;
            padding-left:20px;
            font-family: 'Josefin Sans';
            font-size: 2vw;
            width: 100%;
            height:8%;
            text-align: center;
        }
        .topnav {
  overflow: hidden;
  background-color: #333;
}

.topnav-right a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
  background-color: #ddd;
  color: black;
}

.topnav-right a.active {
  background-color: #565961;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
}

body {

  background-color: #ffffff;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
  .button {
  background-color: #091425;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
```

```
    font-size: 12px;
    border-radius: 16px;
  }
  .button:hover {
    box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
  }
  form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

  input[type=text], input[type=password] {
    width: 100%;
    padding: 12px 20px;
    display: inline-block;
    margin-bottom:18px;
    border: 1px solid #ccc;
    box-sizing: border-box;
  }

  button {
    background-color: #091425;
    color: white;
    padding: 14px 20px;
    margin-bottom:10px;
    border: none;
    cursor: pointer;
    width: 17%;
    border-radius:4px;
    font-family:Montserrat;
  }

  button:hover {
    opacity: 0.8;
  }

  .cancelbtn {
    width: auto;
    padding: 10px 18px;
    background-color: #f44336;
  }

  .imgcontainer {
    text-align: center;
    margin: 24px 0 12px 0;
  }

  img.avatar {
    width: 30%;
    border-radius: 50%;
  }

  .container {
    padding: 16px;
  }

  span.psw {
    float: right;
    padding-top: 16px;
  }

  /* Change styles for span and cancel button on extra small screens */
  @media screen and (max-width: 300px) {
    span.psw {
       display: block;
       float: none;
    }
    .cancelbtn {
       width: 100%;
    }
  }

  .home{
     margin:80px;

    width: 84%;
    height: 500px;
    padding-top:10px;
```

```
    padding: 8px 12px;
    position: absolute;
    bottom: 8px;
    width: 100%;
    text-align: center;
  }
  /* The dots/bullets/indicators */
  .dot {
    height: 15px;
    width: 15px;
    margin: 0 2px;
    background-color: #bbb;
    border-radius: 50%;
    display: inline-block;
    transition: background-color 0.6s ease;
  }

  .active {
    background-color: #717171;
  }

  /* Fading animation */
  .fade {
    -webkit-animation-name: fade;
    -webkit-animation-duration: 1.5s;
    animation-name: fade;
    animation-duration: 1.5s;
  }

  @-webkit-keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
  }

  @keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
  }

  /* On smaller screens, decrease text size */
  @media only screen and (max-width: 300px) {
    .text {font-size: 11px}
  }
  </style>
  </head>

  <body style="font-family:'Algerian', Times, serif;background-color:#ffffff;">

  <div class="header">
  <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-top:1%;padding-left:5%;">Restaurant Recommendation System</div>
    <div class="topnav-right"style="padding-top:0.5%;">

      <a class="active" href="{{ url_for('home')}}">Home</a>
      <a href="{{ url_for('extractor')}}">Recommend</a>
    </div>
  </div>
  <div style="background-color:#ffffff;">
  <div style="width:60%;float:left;">
  <div style="font-size:30px;font-family:Montserrat;padding-left:10px;text-align:center;padding-top:10%;">
  <b>Build recommendation system with ease!!</b></div><br><br>
  <div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-right:30px;text-align:justify;">In this age of information overload, people use a variety of strategies to make choices about what to buy, how to spend their leisure time, and even where to go. Recommender systems automate some of these strategies with
  <div style="font-size:20px;font-family:Montserrat;padding-left:70px;padding-right:30px;"><b>Test the system</b>  
  <a href="{{ url_for('extractor') }}"><button type="submit">Recommend</button></a>

  </div>
  </div>
  <div style="width:40%;float:right;"><br><br>
  <img src="https://media1.giphy.com/media/QVgEz93qOk9mnnU93/giphy.gif?cid=790b7611c457886c75dce80666a818c04d6a71305eed14c8e1d=giphy.gif&ct=g" style="max-height:100%;max-width:100%;">
  </div>
  </div>
  <div class="home">
  <br>
  </div>
  </body>
  </html>
```

Extractor.html code:

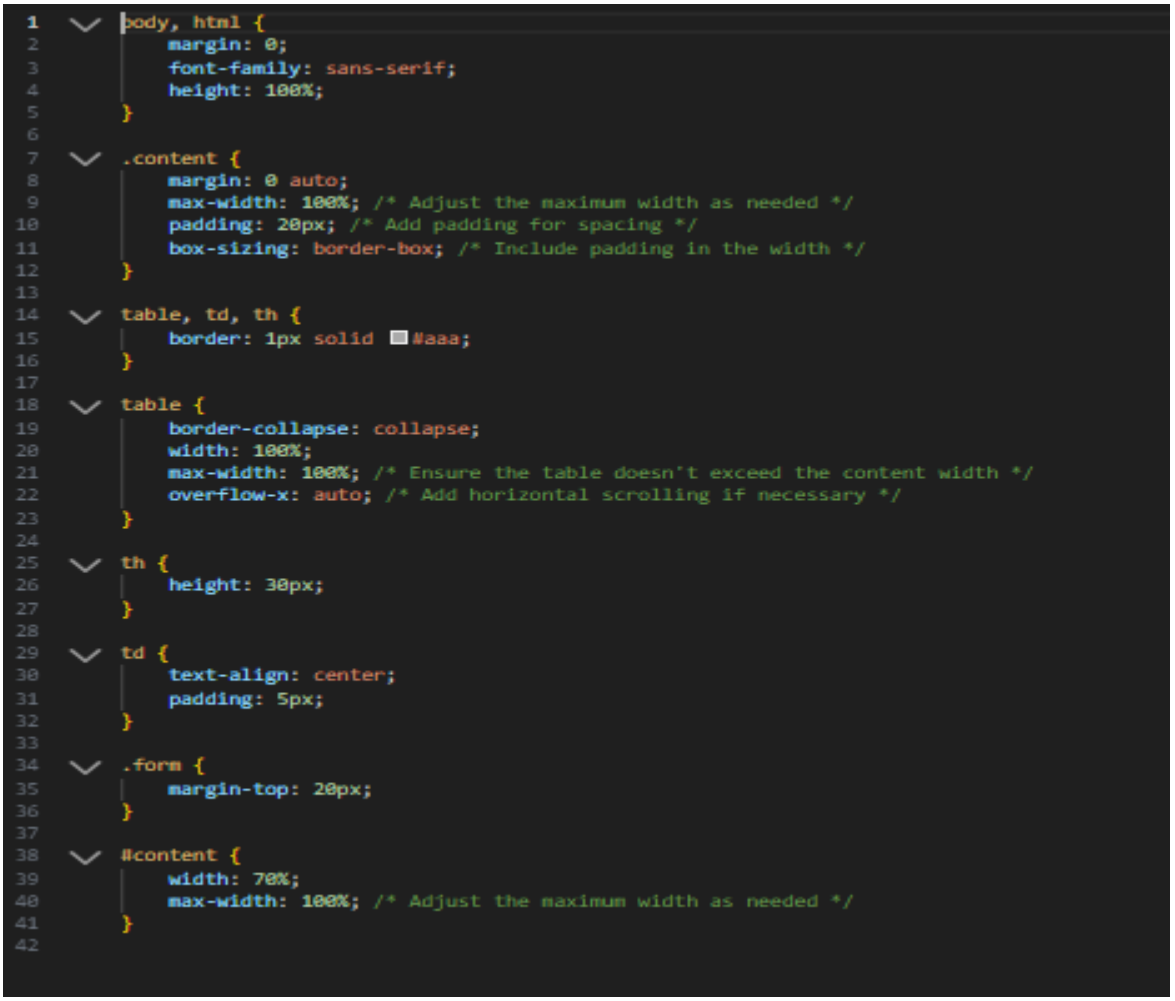
```
body {

  background-color: #ffffff;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
  .button {
  background-color: #091425;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  border-radius: 12px;
}
.button:hover {
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form { margin-left:400px;margin-right:400px;}

select {
    width: 50%;
    margin-bottom: 10px;
    background: rgba(255,255,255,255);
    outline: none;
    padding: 10px;
    font-size:1.2vw;
    color: #000000;
    text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
    border:3px solid #ddd;
    border-radius:4px;
    font-family:Montserrat;
    margin-left: 9%;
}

input[type=text]{
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 3px solid #ddd;
  border-radius:4px;
    font-family:Montserrat;
  box-sizing: border-box;
}

textarea {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 3px solid #ddd;
  border-radius:4px;
  box-sizing: border-box;
  font-family:Montserrat;
}

button {
  background-color: #091425;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 15%;
  border-radius:4px;
  font-family:Montserrat;
}

button:hover {
  opacity: 0.8;
}
```

```
    }

    .cancelbtn {
      width: auto;
      padding: 10px 18px;
      background-color: #f44336;
    }

    .imgcontainer {
      text-align: center;
      margin: 24px 0 12px 0;
    }

    img.avatar {
      width: 30%;
      border-radius: 50%;
    }

    .container {
      padding: 16px;
    }

    span.psw {
      float: right;
      padding-top: 16px;
    }

    /* Change styles for span and cancel button on extra small screens */
    @media screen and (max-width: 300px) {
      span.psw {
         display: block;
         float: none;
      }
      .cancelbtn {
         width: 100%;
      }
    }

    .home{
        margin:80px;

      width: 84%;
      height: 500px;
      padding-top:10px;
      padding-left: 30px;

    }
    ft,.right{
     box-sizing: content-box;
     height: 400px;
     margin:20px;
     border: 10px solid blue;
    }


    /* The dots/bullets/indicators */
    .dot {
      height: 15px;
      width: 15px;
      margin: 0 2px;
      background-color: #bbb;
      border-radius: 50%;
      display: inline-block;
      transition: background-color 0.6s ease;
    }

    .active {
      background-color: #717171;
    }


    /* On smaller screens, decrease text size */
    @media only screen and (max-width: 300px) {
      .text {font-size: 11px}
    }
```

```
</style>
</head>

<body style="font-family:'Times New Roman', Times, serif;background-color:#ffffff;overflow: scroll;">

<div class="header">
 <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-top:1%;padding-left:8%;">Restaurant Recommendation System</div>
  <div class="topnav-right"style="padding-top:0.5%;">

    <a class="active" href="{{ url_for('home')}}">Home</a>
  </div>
</div>

<div style="background-color:#ffffff;">
<div style="width:100%;float:left;">

<div style="font-size:27px;font-family:Montserrat;padding-left:10%;text-align:left;padding-top:1%;">
<b>Give any book name </b>
</div> <br><br>

<form action="{{ url_for('keywords') }}" method="post" style="font-size:20px;font-family:Montserrat;">

  <label for="output">Restaurant Name</label>
  <input type="text" id="output" name="output"><br><br>


</div>

<div style="font-size:20px;font-family:Montserrat;padding-left:9%;padding-right:30px;">
<button type="submit">Click to see the recommendation</button>
</div>
<br><br>
</body>
</html>
```

We also use JavaScript-main.js and CSS-main.css and style.css to enhance our functionality and view of HTML pages.

Main.css Code:



```css
body, html {
    margin: 0;
    font-family: sans-serif;
    height: 100%;
}

.content {
    margin: 0 auto;
    max-width: 100%; /* Adjust the maximum width as needed */
    padding: 20px; /* Add padding for spacing */
    box-sizing: border-box; /* Include padding in the width */
}

table, td, th {
    border: 1px solid #aaa;
}

table {
    border-collapse: collapse;
    width: 100%;
    max-width: 100%; /* Ensure the table doesn't exceed the content width */
    overflow-x: auto; /* Add horizontal scrolling if necessary */
}

th {
    height: 30px;
}

td {
    text-align: center;
    padding: 5px;
}

.form {
    margin-top: 20px;
}

#content {
    width: 70%;
    max-width: 100%; /* Adjust the maximum width as needed */
}
```

Style.css Code:

```css
body {
  background-color: #91ced4;
}
body * {
  box-sizing: border-box;
}

.header {
  background-color: #327a81;
  color: white;
  font-size: 1.5em;
  padding: 1rem;
  text-align: center;
  text-transform: uppercase;
}

img {
  border-radius: 50%;
  height: 60px;
  width: 60px;
}

.table-users {
  border: 1px solid #327a81;
  border-radius: 10px;
  box-shadow: 3px 3px 0 rgba(0, 0, 0, 0.1);
  max-width: calc(100% - 2em);
  margin: 1em auto;
  overflow: hidden;
  width: 800px;
}

table {
  width: 100%;
}
table td, table th {
  color: #2b686e;
  padding: 10px;
}
table td {
  text-align: center;
  vertical-align: middle;
}
table td:last-child {
  font-size: 0.95em;
  line-height: 1.4;
  text-align: left;
}
table th {
  background-color: #daeff1;
  font-weight: 300;
}
table tr:nth-child(2n) {
  background-color: white;
}
table tr:nth-child(2n+1) {
  background-color: #edf7f8;
}

@media screen and (max-width: 700px) {
  table, tr, td {
    display: block;
  }

  td:first-child {
    position: absolute;
    top: 50%;
    -webkit-transform: translateY(-50%);
            transform: translateY(-50%);
    width: 100px;
  }
  td:not(:first-child) {
    clear: both;
    margin-left: 100px;
    padding: 4px 20px 4px 90px;
    position: relative;
```

```
    }
    td:nth-child(3):before {
      content: 'Email:';
    }
    td:nth-child(4):before {
      content: 'Phone:';
    }
    td:nth-child(5):before {
      content: 'Comments:';
    }

    tr {
      padding: 10px 0;
      position: relative;
    }
    tr:first-child {
      display: none;
    }
  }
  @media screen and (max-width: 500px) {
    .header {
      background-color: transparent;
      color: white;
      font-size: 2em;
      font-weight: 700;
      padding: 0;
      text-shadow: 2px 2px 0 rgba(0, 0, 0, 0.1);
    }

    img {
      border: 3px solid;
      border-color: #daeff1;
      height: 100px;
      margin: 0.5rem 0;
      width: 100px;
    }

    td:first-child {
      background-color: #c8e7ea;
      border-bottom: 1px solid #91ced4;
      border-radius: 10px 10px 0 0;
      position: relative;
      top: 0;
      -webkit-transform: translateY(0);
              transform: translateY(0);
      width: 100%;
    }
    td:not(:first-child) {
      margin: 0;
      padding: 5px 1em;
      width: 100%;
    }
    td:not(:first-child):before {
      font-size: .8em;
      padding-top: 0.3em;
      position: relative;
    }
    td:last-child {
      padding-bottom: 1rem !important;
    }

    tr {
      background-color: white !important;
      border: 1px solid #6cbec6;
      border-radius: 10px;
      box-shadow: 2px 2px 0 rgba(0, 0, 0, 0.1);
      margin: 0.5rem 0;
      padding: 0;
    }

    .table-users {
      border: none;
      box-shadow: none;
      overflow: visible;
    }
  }
```

Activity 2: Build python code

● We'll create a Flask file named 'app1.py', a Python-based web framework for server-side scripting. Here's a step-by-step guide to developing the backend application.

● The application initiates when the "name" constructor is invoked in the main section.

● 'Render template' is employed to send back an HTML file.

● The "GET" method is utilized for user input.

● The "POST" method is employed to present output to the user.

Importing libraries

C: > Users > DELL > AppData > Local > Temp > 5ebadc44-9128-4c23-970a-8dab4406f884_restaurant2.zip.884 > restaurant2 > flask > app1.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

```

Libraries required for the app to run are to be imported.

Creating our flask app and loading the newly created dataset

Now after all the libraries are import we will be creating our flask app with the updated dataset

```

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")
```

Routing to the html Page:

We use page routes to display the user interface (UI) of our HTML pages, linking the built code in these pages to our Flask application. This method allows us to create and demonstrate the UI by connecting it to our backend Flask app.

```
27
28  @app.route('/')# route to display the home page
29  def home():
30      return render_template('home.html')#rendering the home page
31
32
33  @app.route('/extractor')
34  def extractor():
35      return render_template('extractor.html')
36
37  #extractor page
```

We're directing the application to the HTML templates we aim to display. Initially, we render the 'home.html' template and from there, we navigate to our prediction page, which is 'indexnew.html'.

```
39
40  @app.route('/keywords',  methods=['POST'])
41  def keywords():
42      output = request.form['output']
43
44      print(output)
45      print(type(output))
46
47      df_percent = zomato_df.sample(frac=0.5)
48      df_percent.set_index('name', inplace=True)
49      indices = pd.Series(df_percent.index)
50
51      tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=1, stop_words='english')  # Change min_df to 1
52      tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'].fillna(' '))
53      cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
54
55      def recommend(name, cosine_similarities=cosine_similarities):
56          recommend_restaurant = []
57          idx = indices[indices == name].index[0]
58          score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False)
59          top30_indexes = list(score_series.iloc[0:31].index)
60
61          for each in top30_indexes:
62              recommend_restaurant.append(list(df_percent.index)[each])
63
64          df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost','url'])
65
66          for each in recommend_restaurant:
67              df_new = pd.concat([df_new, df_percent[['cuisines','Mean Rating', 'cost','url']][df_percent.index == each].sample()])
68
69          df_new = df_new.drop_duplicates(subset=['cuisines','Mean Rating', 'cost','url'], keep=False)
70          pd.set_option('display.max_columns', None)
71
72          df_new = df_new.sort_values(by='Mean Rating', ascending=False).head(10)  # Fix the sorting line
73          print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' % (str(len(df_new)), name))
74
75          return df_new
76
77      result = recommend(output)
78      print(result)
79      print(type(result))
80
81      return render_template('keywords.html', keyword=result.to_html())
82
```

Lastly, we run our app on the local host.

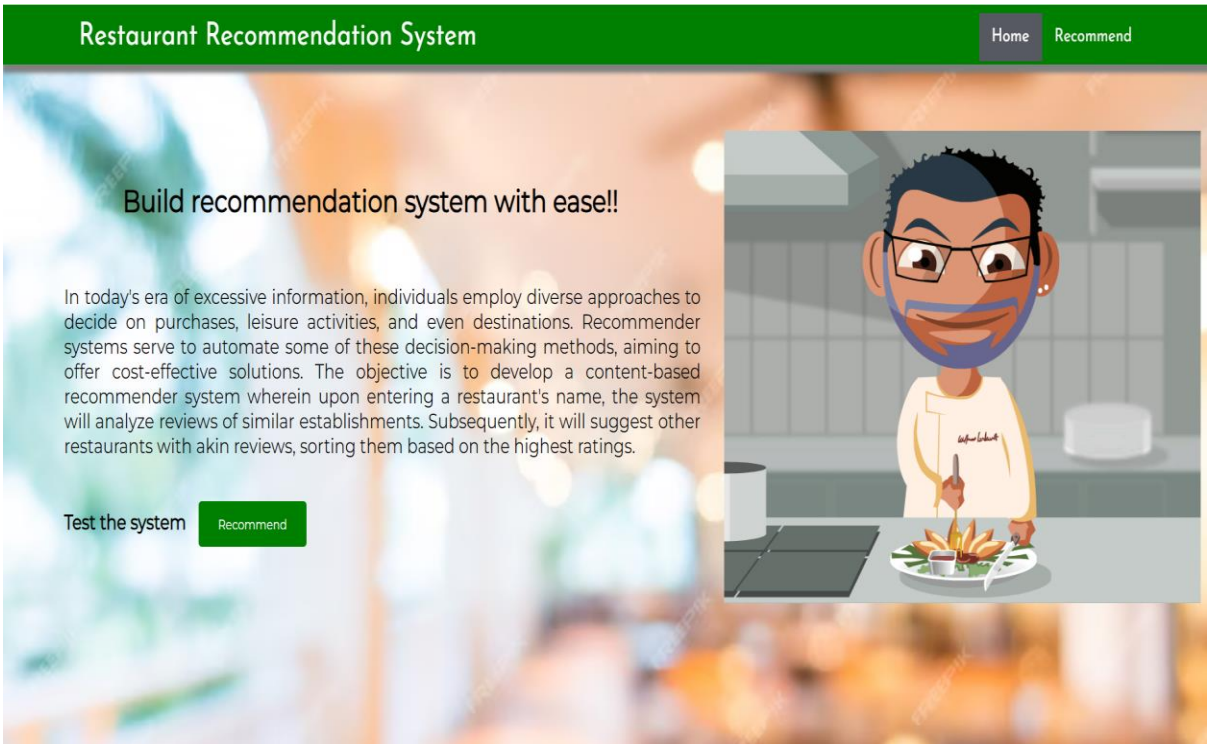
```
83      if __name__ == "__main__":
84          # running the app
85           app.run(debug=True)
86
```

Here we are running it on localhost:5000

Activity:3 Run The app in local browser

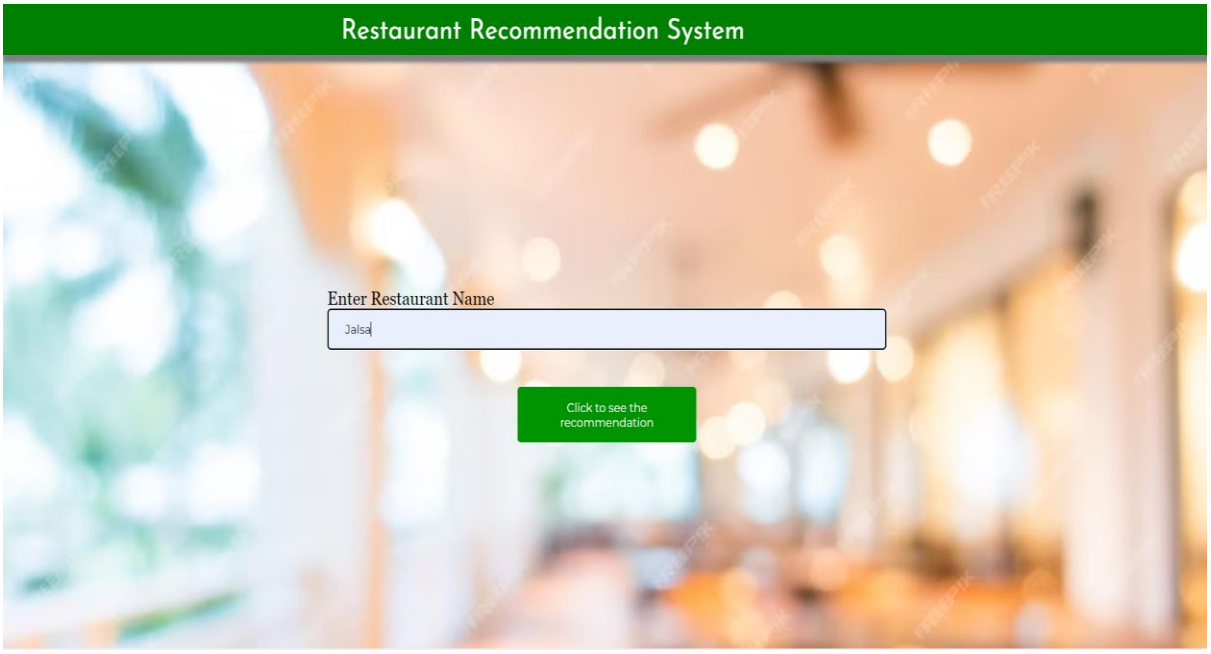
```
ion_system> python app1.py
 * Serving Flask app 'app1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 118-588-047
127.0.0.1 - - [04/Nov/2023 00:14:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Nov/2023 00:14:47] "GET /favicon.ico HTTP/1.1" 404 -
```

- Launch Anaconda Prompt from the start menu.
- Go to the directory where your Python script is located.
- Enter the command "python app.py" in the prompt.
- Access the localhost to view your web page.

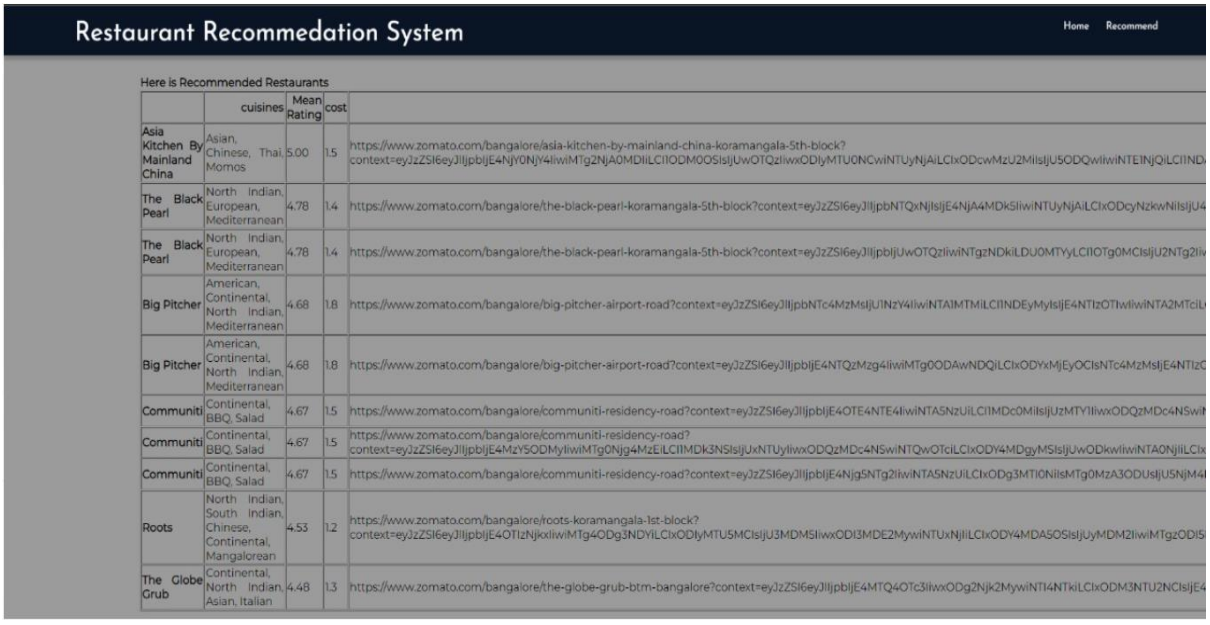


This is the home main page that describes the project and summarizes it.

I. Checking recommendation for the restaurant: ‘Jalsa’

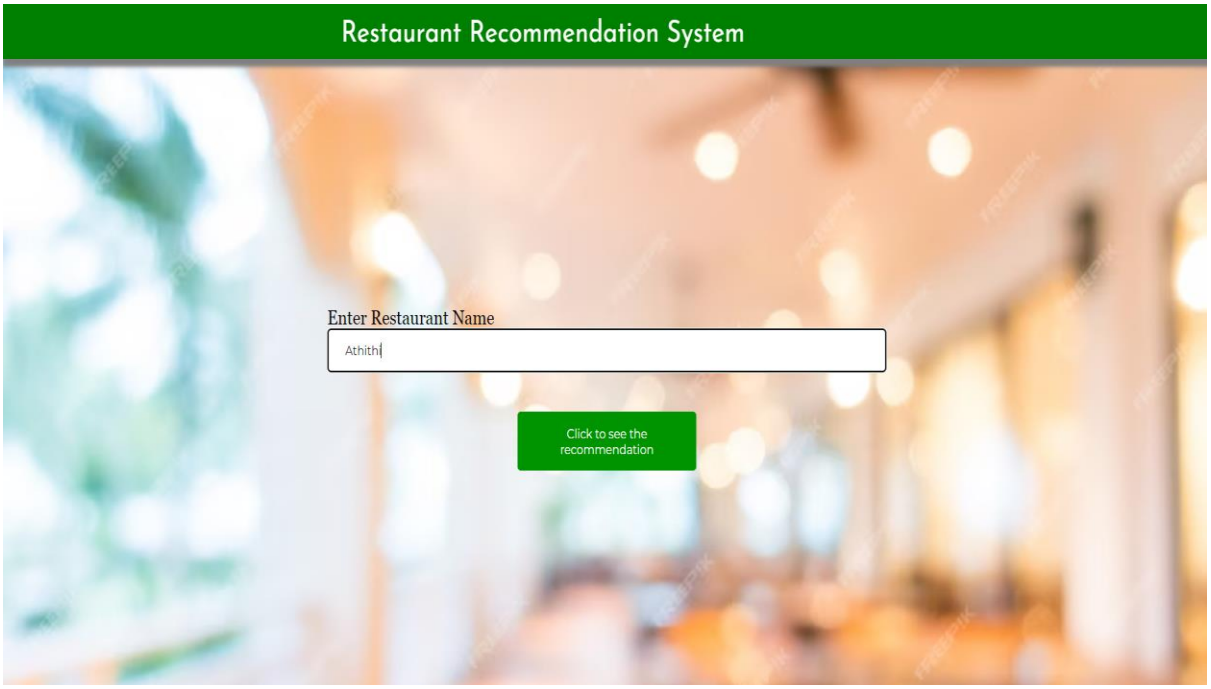


This is the page for predictions, where we input a restaurant's name to receive the top recommended restaurants. Recommendations are based on factors like cuisines, average rating (on a scale of 5), and cost in thousands.

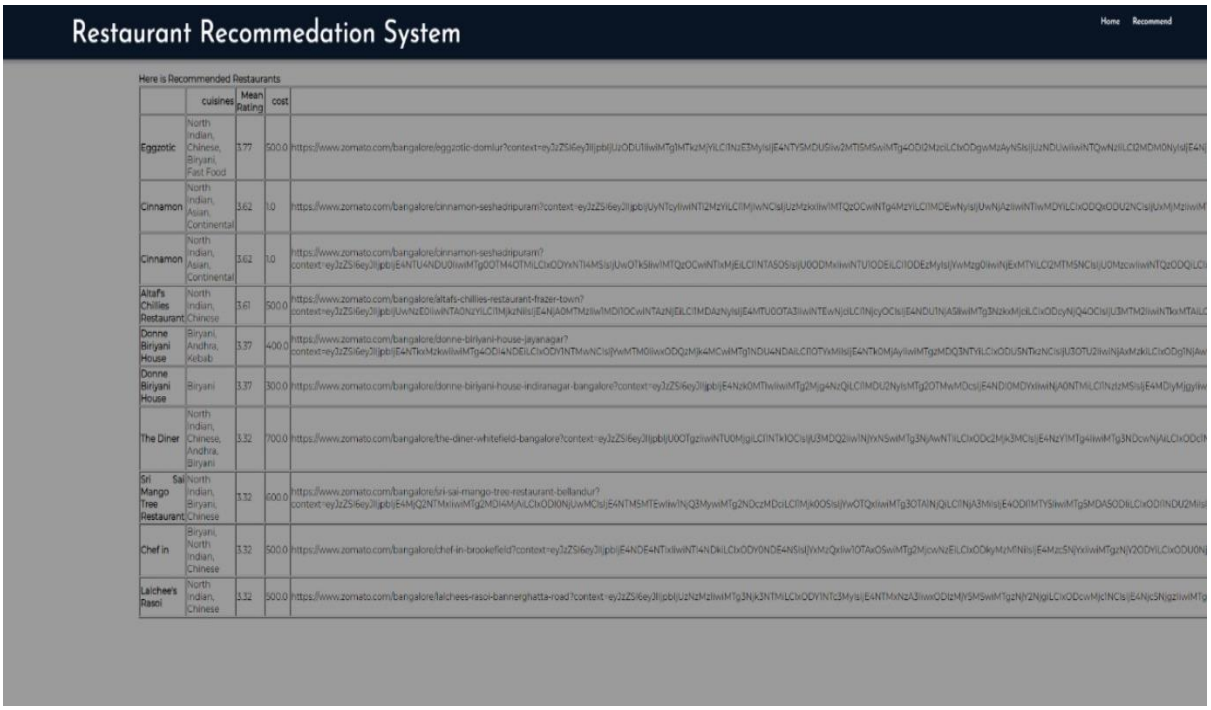


	cuisines	Mean Rating	cost	
Asia Kitchen By Mainland China	Asian, Chinese, Thai, Momos	5.00	1.5	https://www.zomato.com/bangalore/asia-kitchen-by-mainland-china-koramangala-5th-block?context=eyJzZSI6eyJlIjpbIjE4NjY0NjY4IiwiMTg2NjA0MDIiLCI1ODM0OSIsIjUwOTQzIiwxODIyMTU0NCwiNTUyNjAiLCIxODcwMzU2MiIsIjU5ODQwIiwiNTE1NjQiLCI1NDA
The Black Pearl	North Indian, European, Mediterranean	4.78	1.4	https://www.zomato.com/bangalore/the-black-pearl-koramangala-5th-block?context=eyJzZSI6eyJlIjpbNTQxNjIsIjE4NjA4MDk5IiwiNTUyNjAiLCIxODcyNzkwNiIsIjU4
The Black Pearl	North Indian, European, Mediterranean	4.78	1.4	https://www.zomato.com/bangalore/the-black-pearl-koramangala-5th-block?context=eyJzZSI6eyJlIjpbIjUwOTQzIiwiNTgzNDkiLDU0MTYyLCI1OTg0MCIsIjU2NTg2Iiw
Big Pitcher	American, Continental, North Indian, Mediterranean	4.68	1.8	https://www.zomato.com/bangalore/big-pitcher-airport-road?context=eyJzZSI6eyJlIjpbNTc4MzMsIjU1NzY4IiwiNTA1MTMiLCI1NDEyMyIsIjE4NTIzOTIwIiwiNTA2MTciL
Big Pitcher	American, Continental, North Indian, Mediterranean	4.68	1.8	https://www.zomato.com/bangalore/big-pitcher-airport-road?context=eyJzZSI6eyJlIjpbIjE4NTQzMzg4IiwiMTg0ODAwNDQiLCIxODYxMjEyOCIsNTc4MzMsIjE4NTIzO
Communiti	Continental, BBQ, Salad	4.67	1.5	https://www.zomato.com/bangalore/communiti-residency-road?context=eyJzZSI6eyJlIjpbIjE4OTE4NTE4IiwiNTA5NzUiLCI1MDc0MiIsIjUzMTY1IiwxODQzMDc4NSwiN
Communiti	Continental, BBQ, Salad	4.67	1.5	https://www.zomato.com/bangalore/communiti-residency-road?context=eyJzZSI6eyJlIjpbIjE4MzY5ODMyIiwiMTg0Njg4MzEiLCI1MDk3NSIsIjUxNTUyIiwxODQzMDc4NSwiNTQwOTciLCIxODY4MDgyMSIsIjUwODkwIiwiNTA0NjIiLCIx
Communiti	Continental, BBQ, Salad	4.67	1.5	https://www.zomato.com/bangalore/communiti-residency-road?context=eyJzZSI6eyJlIjpbIjE4Njg5NTg2IiwiNTA5NzUiLCIxODg3MTI0NiisMTg0MzA3ODUsIjU5NjM4
Roots	North Indian, South Indian, Chinese, Continental, Mangalorean	4.53	1.2	https://www.zomato.com/bangalore/roots-koramangala-1st-block?context=eyJzZSI6eyJlIjpbIjE4OTIzNjkxIiwiMTg4ODg3NDYiLCIxODIyMTU5MCIsIjU3MDMSIiwxODI3MDE2MywiNTUxNjIiLCIxODY4MDA5OSIsIjUyMDM2IiwiMTgzODI5
The Globe Grub	Continental, North Indian, Asian, Italian	4.48	1.3	https://www.zomato.com/bangalore/the-globe-grub-btm-bangalore?context=eyJzZSI6eyJlIjpbIjE4MTQ4OTc3IiwxODg2Njk2MywiNTI4NTkiLCIxODM3NTU2NCIsIjE4

II. Checking recommendation for the restaurant ‘Athithi’



And here is the recommendation



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