RESTAURANT RECOMMENDATION SYSTEM

Date	15 November 2023
Team ID	592046
Project Name	Restaurant recommendation system

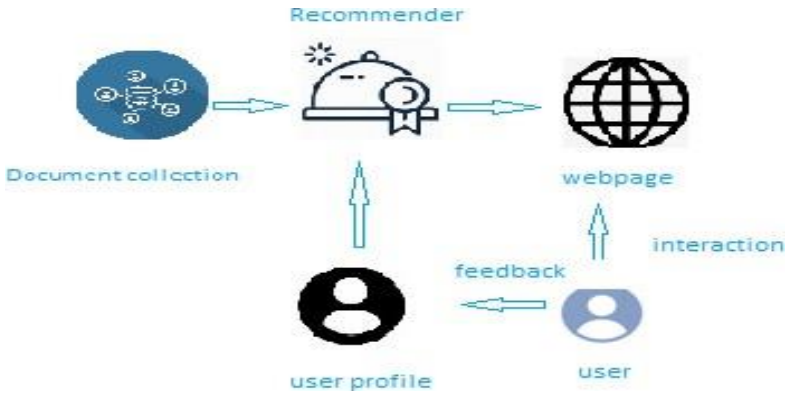
PROJECT DESCRIPTION

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.

SOLUTION

Here we are creating a content-based recommendation system. The aim is to create a content-based recommender system in which when we will write a restaurant name, the Recommender system will **look at the reviews of other restaurants**, and the System will recommend us other restaurants with **similar reviews** and sort them from the **highest-rated**. The main people who are going to benefit from this recommendation system are the tourists, who are new to a city. Most of the tourists always love to visit famous restaurants in a particular city during their visit. Otherwise, it can be heavily used by people belonging to the same city, to see if any new restaurant is recommended based on their activity.

ARCHITECTURE


Recommender
Documcnt collcction
webpage
interaction
feedback
user profile
user

LEARNING OUTCOMES

By the end of this project:

- You'll be able to perform one of the techniques to build you recommendation system
- You'll be able to know the recommendation system using Content-Based Filtering.
- You will be able to know how to pre-process / clean the data using different data pre-processing techniques.
- You will able to analyse or get insights of data through visualization.
- Applying algorithms according to dataset and based on visualization.
- You will able to know how to find accuracy of the model.
- You will be able to know how to build a web application using Flask framework.

PRE-REQUISITES:

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

Activity 1: Install Anaconda IDE / Anaconda Navigator.

- In order develop a solution to this problem statement, we need an environment to write and test the code.
- We use Anaconda IDE (Integrated Developing Environment).
- Refer to the below link to download & install Anaconda Navigator.

Link: https://www.youtube.com/watch?v=5mDYijMfSzs

1. **Activity 2: To build Machine learning models you must require the following packages**

- **Numpy**:

  It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations

- **Numpy**:

  It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy

- **Matplotlib and Seaborn**

  Matplotlib is mainly deployed for basic plotting. Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on. Seaborn: Seaborn, on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

- **Flask:**

  Web framework used for building Web applications

If you are using **anaconda navigator**, follow below steps to download required packages:

- Open anaconda prompt.
- 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.

Link: Introduction to Scikit-Learn (sklearn) in Python • datagy

PRIOR KNOWLEDGE

One should have knowledge on the following Concepts:

Link: Supervised and Unsupervised Learning

Watch the below video to know about the types of machine learning

Link: Regression, Classification and Clustering

Link: ML - Content Based Recommender System - GeeksforGeeks

Link: NLTK :: Natural Language Toolkit

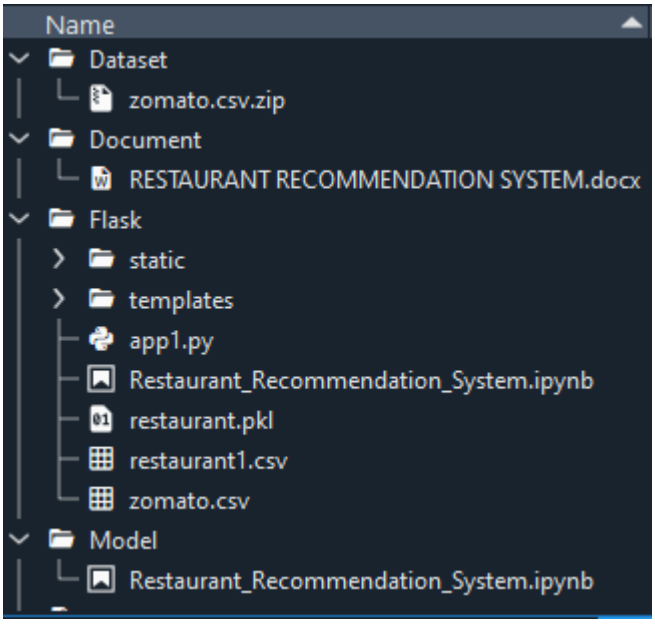
Link: Flask:

Link: Recommendation System

It is recommended to watch above video's to understand the concepts before you start your project.

PROJECT WORK FLOW

- User interacts with the UI (User Interface) to enter the input features.
- Entered features/input is analysed by the model which is integrated
- Once model analyses the entered inputs, the prediction is showcased on the UI.



TASKS

1. Data Collection.
   - Collect the dataset or Create the dataset

2. Data Pre- processing.
    - Import the Libraries.
    - Importing the dataset.
    - Exploratory Data Analysis
    - Data Visualization.

3. Content Based Filtering
    - Merging datasets
    - Creating the recommender system
    - Predicting the results

4. Application Building
    - Create an HTML file
    - Build a Python Code

Milestone 1: Data Collection

Now, the milestone-2 is all about creation or collection of dataset.

we will use the **Zomato** Bangalore for our analysis to draw conclusions using the content filtering method.

Here is the dataset link: **dataset**

Milestone 2: Data Pre-processing

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

Activity 1: Import Libraries

Import the below essential libraries for data pre-processing and creating recommendation system. Pandas and NumPy are used for data pre-processing and cleaning. Seaborn, Plotly and Matplotlib helped in creating visual graphics and bar plots for the dataset. Also, since there would be cleaning of text data (reviews) as well, therefore for that we will use nltk and sklearn library.

```
In [2]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import plotly.offline as py
        import plotly.graph_objs as go

        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:

Our dataset format might be in .csv, excel files, .txt, json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv () to read the dataset. As a parameter we have to give the directory of csv file.

```
In [11]: zomato_data=pd.read_csv("zomato.csv")

In [12]: zomato_data
```

Out[12]:

	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type
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
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
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashankari	Cafe, Casual Dining
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashankari	Quick Bites

Activity 3: Analyse the Dataset:

```
In [16]: zomato_df.shape
Out[16]: (51717, 17)
```

The dataset contains 51717 records with 17 features.

Checking the columns in the dataset.

```
In [17]: zomato_df.columns
Out[17]: Index(['url', 'address', 'name', 'online_order', 'book_table', 'rat
         e', 'votes',
                'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
                'approx_cost(for two people)', 'reviews_list', 'menu_item',
                'listed_in(type)', 'listed_in(city)'],
               dtype='object')
```

Columns description

1. **URL** contains the url of the restaurant on the Zomato website
2. **address** contains the address of the restaurant in Bengaluru

3. name contains the name of the restaurant

4. online_order whether online ordering is available in the restaurant or not

5. book_table table book option available or not

6. rate contains the overall rating of the restaurant out of 5

7. votes contain the total number of rating for the restaurant as of the above-mentioned date

8. phone contains the phone number of the restaurant

9. location contains the neighbourhood in which the restaurant is located

10. rest_type restaurant type

11. dish_liked dishes people liked in the restaurant

12. cuisines food styles, separated by comma

13. approx_cost(for two people) contains the approximate cost for a meal for two people

14. reviews_list list of tuples containing reviews for the restaurant, each tuple

15. menu_item contains a list of menus available in the restaurant

16. listed_in(type) type of meal

17. listed_in(city) contains the neighbourhood in which the restaurant is listed.

Understanding Overview of features

- How the information is stored in a DataFrame or Python object affects what we can do with it and the outputs of calculations as well. There are two main types of data those are numeric and text data types.
- Numeric data types include integers and floats.
- Text data type is known as Strings in Python, or Objects in Pandas. Strings can contain numbers and / or characters.
- For example, a string might be a word, a sentence, or several sentences.
- Will see how our dataset is, by using **info ()** method.

```
In [18]: 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
```

- As you can see in our dataset, except ‘votes’, all other features are categorical data, but it is not necessary that all the continuous data which we are seeing has to be continuous in nature. There may be a case that some categorical data is in the form of numbers but when we perform info () operation we will get numerical output. So, we need to take care of those type of data also.

Checking for null values in the dataset

```
In [19]: zomato_df.isnull().sum()
```

```
Out[19]: 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

```
In [20]: zomato_df=zomato_df.drop(['phone','dish_liked','url'],axis=1)
```

```
In [21]: zomato_df.dropna(how='any',inplace=True)
```

```
In [23]: zomato_df.duplicated().sum()
         zomato_df.drop_duplicates(inplace=True)
```

```
In [24]: zomato_df = zomato_df.rename(columns={'approx_cost(for two people)':
```

```
In [31]: import numpy as np
```

```
In [32]: zomato_df = zomato_df[zomato_df['rate'] != 'NEW']
```

```
In [33]: omato_df = zomato_df[zomato_df['rate'] != '-'].reset_index(drop=True)
```

```
In [34]: da x: float(x.replace('/5', '').strip()) if type(x) == np.str else x
```

```
In [35]: zomato_df['rate'] = zomato_df['rate'].apply(remove_slash)
```

In [36]:
```
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)
```

Checking for null values after cleaning & data Processing

```
In [37]: zomato_df.isnull().sum()
```

```
Out[37]: 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
```

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

```
In [34]: zomato_df[['name','rate','Mean Rating']].head()
```

Out[34]:

	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

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

```
In [35]: zomato_df["reviews_list"] = zomato_df["reviews_list"].str.lower()

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

Out[36]:

	reviews_list	cuisines
40587	rated 30 ratedn kaayal is our neighbourhood k...	Kerala, Chinese
20122	rated 40 ratedn its a hidden gem in the area ...	Salad, Chinese
7285	rated 40 ratedn ordered beef biryani chicken...	Kerala, South Indian, Biryani
17274	rated 10 ratedn the only thing worse than the...	Cafe, Italian, Desserts
22956	rated 40 ratedn i ordered a keto chicken tikk...	Healthy Food

Milestone 3: Data Visualization

Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data. Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.

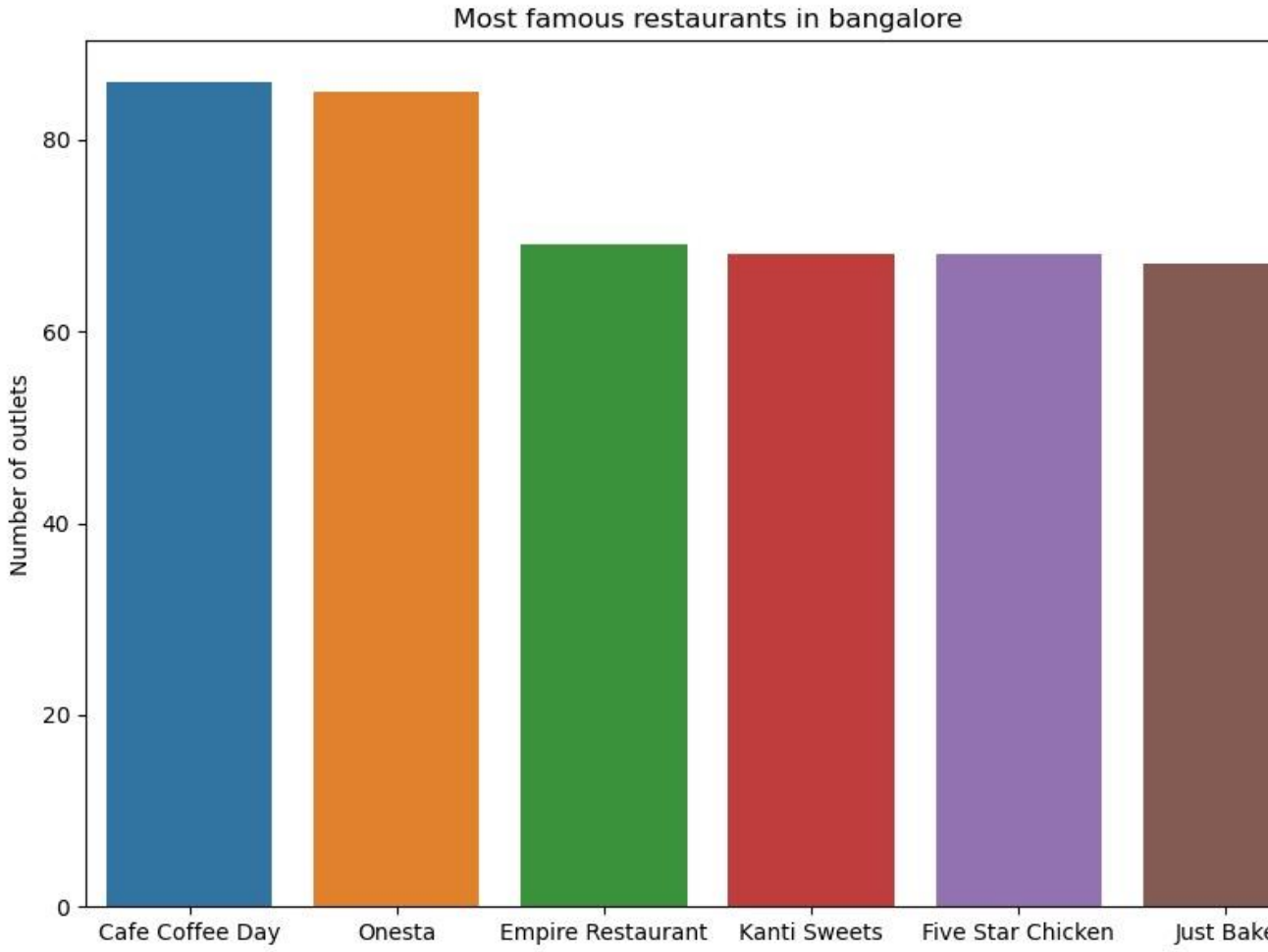
To visualize the dataset, we need libraries called Matplotlib, Seaborn. The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc.

Let's visualize our data using Matplotlib and seaborn library.

At first, we will be plotting a bar plot using matplotlib for showing the top 6 restaurants in Bangalore by value counts.

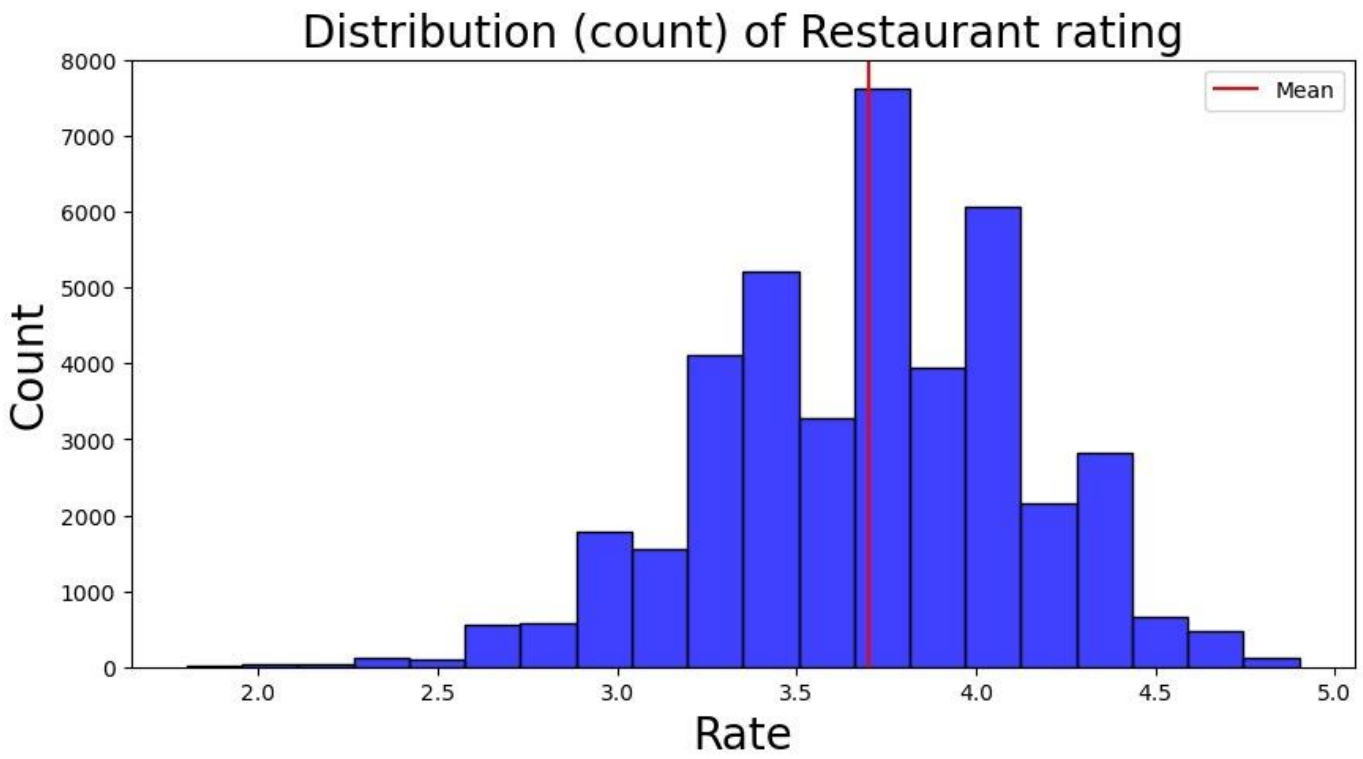
```
In [37]: #Most Famous 6 restaurants in bangalore

         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")
```

```
Out[37]: Text(0, 0.5, 'Number of outlets')
```



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

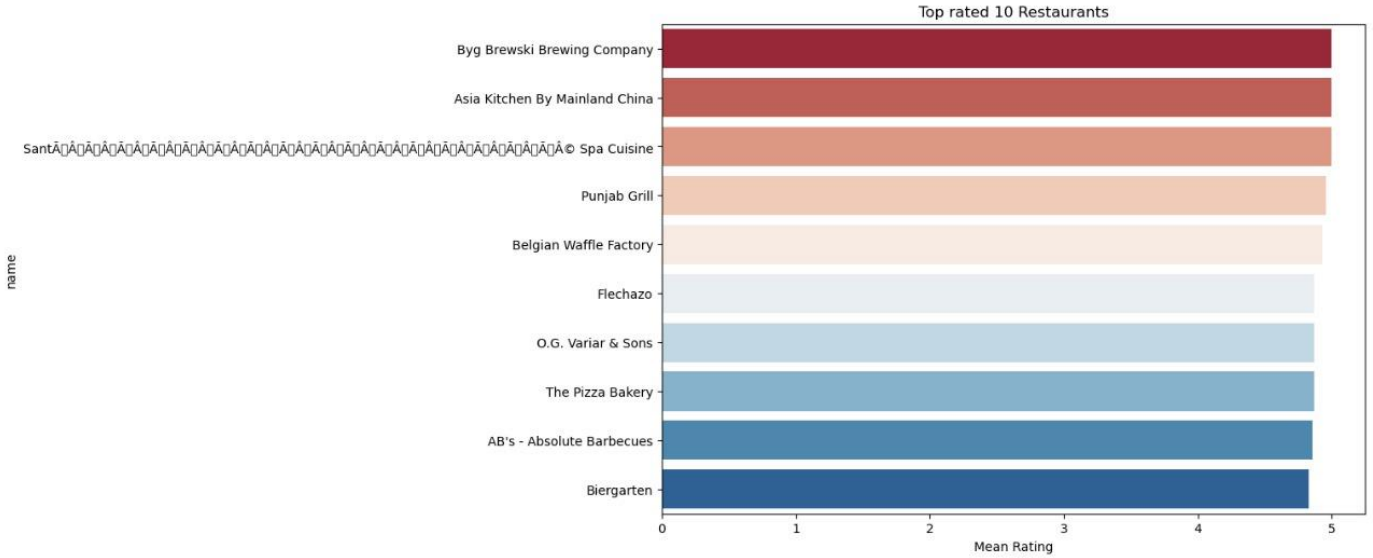
```
In [40]: # Distribution of Restaurant Rating
         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10, 5))
         sns.histplot(zomato_df['rate'], kde=False, color='b', ax=ax, bins=20)
         ax.axvline(zomato_df['rate'].mean(), 0, 1, color='r', label='Mean')
         ax.legend()
         ax.set_ylabel('Count', size=20)
         ax.set_xlabel('Rate', size=20)
         ax.set_title('Distribution (count) of Restaurant rating', size=20)
         plt.show()
```



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.

In [41]:
```
#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');
```

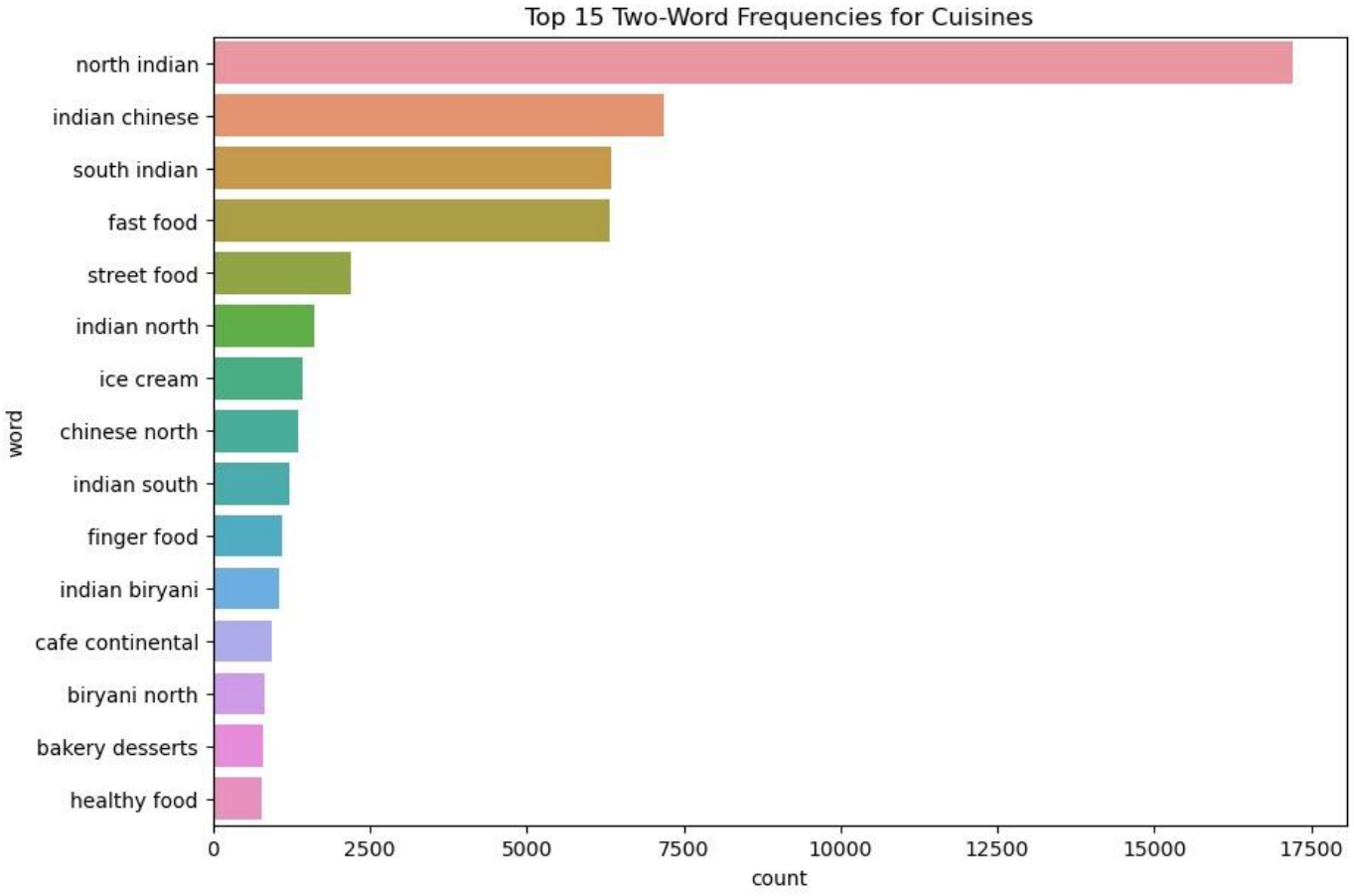


Visualizing two word frequencies for cuisines, using barplot from seaborn library.

In [43]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
```

In [44]:
```
def get_top_words(text_series, n, ngram_range):
    vectorizer = CountVectorizer(ngram_range=ngram_range)
    X = vectorizer.fit_transform(text_series)
    word_counts = X.sum(axis=0)
    word_counts = [(word, word_counts[0, idx]) for word, idx in vectorizer.vocabulary_.items()]
    word_counts = sorted(word_counts, key=lambda x: x[1], reverse=True)
    return word_counts[:n]
```

In [45]:
```
# 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('Top 15 Two-Word Frequencies for Cuisines')
plt.show()
```



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)
```

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

```
In [*]: import pandas as pd

def recommend(name, cosine_similarities, indices, df_percent):
    # Create a list to put top restaurants
    recommend_restaurant = []

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

    # Find the restaurants with a similar cosine-sim value and order them from the biggest score
    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 dataset to show similar restaurants
    df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost'])

    # 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']][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'], keep='first')
    df_new = df_new.sort_values(by='Mean Rating', ascending=False).head(10)

    print(f"TOP {len(df_new)} RESTAURANTS LIKE {name} WITH SIMILAR REVIEWS:")
    print(df_new)

    return df_new

# Example usage:
# Assuming you have cosine_similarities, indices, and df_percent already defined
recommendations = recommend('YourRestaurantName', cosine_similarities, indices, df_percent)
```

Share this windo

Querying recommendation for 4 Restaurants:

For Restaurant ‘Red Chilliez’

```
recommend('Jalsa')
```

```
TOP 9 RESTAURANTS LIKE Jalsa WITH SIMILAR REVIEWS:
```

	cuisines	Mean Rating	cost
Vapour Brewpub And Diner	North Indian, Continental, Italian	4.54	1.4
Roots	North Indian, South Indian, Chinese, Continent...	4.53	1.2
The Globe Grub	Continental, North Indian, Asian, Italian	4.48	1.3
The Pallet	Continental, Mediterranean, Italian, North Ind...	4.48	1.6
Btdt? Been There Done That	European, Continental, Asian, Finger Food	4.23	1.3
Jalsa	North Indian, Mughlai, Chinese	3.99	800.0
Jalsa	North Indian, Mughlai	3.99	1.5
Urban Tamaasha	North Indian, Chinese, Continental	3.97	1.2
Cinnamon	North Indian, Asian, Continental	3.62	1.0

For Restaurant 'Cinnamon':

```
recommend('Red Chilliez')
```

```
TOP 10 RESTAURANTS LIKE Red Chilliez WITH SIMILAR REVIEWS:
```

	cuisines	Mean Rating	cost
Pallavi Restaurant	Biryani, Chinese, Andhra	3.58	500.0
B.M.W - Bhookh Mitaane Wala	North Indian, South Indian, Chinese	3.42	500.0
Agarwal Food Service	North Indian, Chinese, Biryani	3.39	400.0
Red Chilliez	North Indian, Chinese, Seafood, Mangalorean	3.26	650.0
Desi Dhaba	North Indian, Chinese	3.19	400.0
Bangalore Bytes	Fast Food, South Indian, Biryani	3.19	300.0
Punjabi Tasty Khana	North Indian, Chinese, Biryani	2.68	450.0
Taza Khaana	Chinese, North Indian	2.63	450.0
Sri Lakshmi Dhaba	North Indian, Chinese	2.50	250.0
Night Food Joint	North Indian, Chinese	2.35	500.0

For Restaurant 'Spice up':

```
recommend('Grand Village')
```

```
TOP 9 RESTAURANTS LIKE Grand Village WITH SIMILAR REVIEWS:
```

	cuisines	Mean Rating	cost
The Black Pearl	North Indian, European, Mediterranean	4.78	1.4
Deja Vu Resto Bar	North Indian, Italian	4.35	900.0
Gramin	North Indian	4.23	600.0
Amritsari Kulcha Land	North Indian	3.86	300.0
Village - The Soul Of India	North Indian, South Indian	3.85	1.0
Atithi	North Indian	3.63	750.0
Grand Village	North Indian, Rajasthani	3.58	600.0
Nouvelle Garden	North Indian, Continental, Italian	3.45	900.0
Thindi Palace	North Indian, South Indian, Chinese	2.68	600.0

For Restaurant 'Desi Doze':

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

```
TOP 9 RESTAURANTS LIKE Cinnamon WITH SIMILAR REVIEWS:
```

	cuisines	Mean Rating	cost
Gokul Kuteera	North Indian, Chinese	3.71	650.0
Gokul Kuteera	North Indian, Chinese, South Indian	3.71	650.0
Atithi	North Indian, Chinese, Street Food	3.63	800.0
Altaf'S Chillies Restaurant	North Indian, Chinese	3.61	500.0
Fazzito Kitchen	North Indian	3.58	700.0
Chef In	Biryani, North Indian, Chinese	3.32	500.0
Aaranya Restaurant	North Indian, Chinese, Mughlai	3.06	800.0
Wazir'S	North Indian, Chinese	2.94	500.0
Night Food Joint	North Indian, Chinese	2.35	500.0

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 2 html pages- index.html, web.html.

index.html displays home page.

web.html accepts the values from the input and displays the prediction.

For more information regarding HTML refer the link below

https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp

- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
    - **Link :https://www.w3schools.com/css/**
    - **https://www.w3schools.com/js/DEFAULT.asp**

Activity 2: Build python code

- Let us build flask file 'app1.py' which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application.
- App starts running when "__name__" constructor is called in main.
- render template is used to return html file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.

Importing libraries

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
import re
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
```

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

```
4     app = Flask(__name__)
5
6     # Load the dataset
7     restaurant_data = pd.read_csv("restaurant1.csv")
8
```

Routing to the html Page:

Basically, we give routes of our html pages in order to show case the UI. By giving the routes the built code in the html page is connected to our flask app. This is how a UI can be built and showcased.

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

@app.route('/recommendation', methods=['POST'])
def recommendation():
    # Get user input from the form
    user_preference = request.form.get('preference')

    # Recommendation logic (Replace this with your actual recommendation logic)
    recommended_restaurants = recommend_restaurants(user_preference)

    return render_template('recommendation.html', user_preference=user_preference, recommendations=recommended_restaurants)
```

We are routing the app to the html templates which we want to render.

Firstly, we are rendering the home.html template and from there we are navigating to our prediction page that is indexnew.html

```python
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)

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'])

    # 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']][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'], 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
recommend('cinnamon')
```

Lastly, we run our app on the local host.

```python
if __name__ == '__main__':
    app.run(debug=True)
```

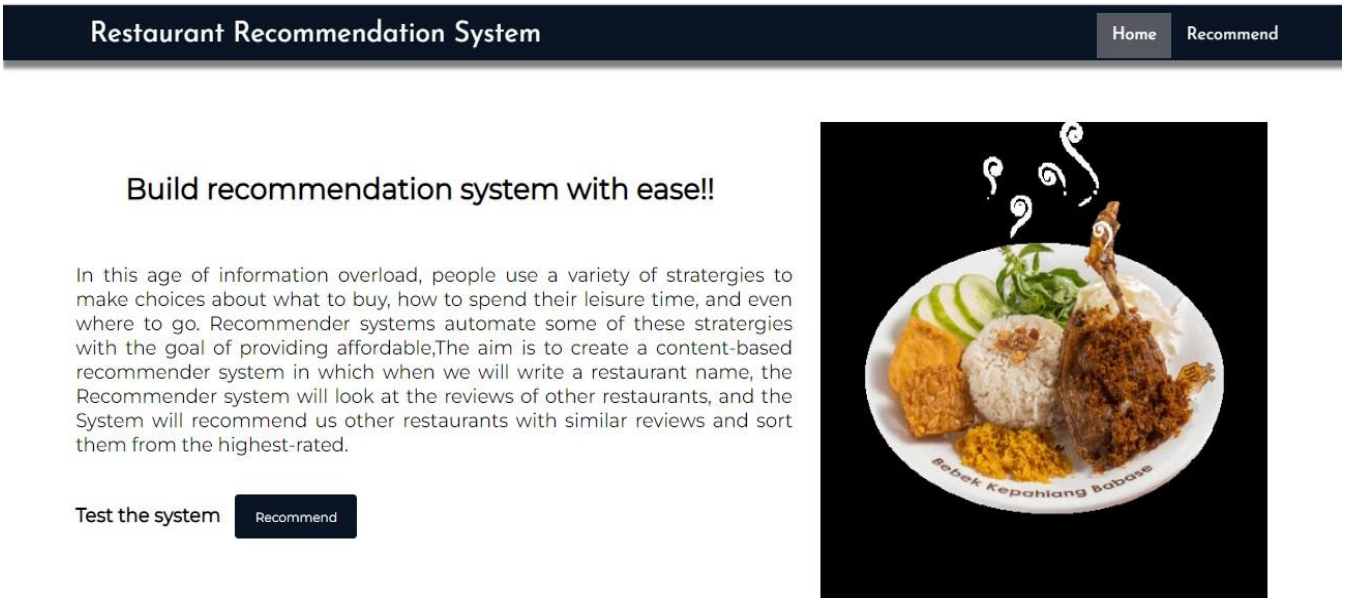
Here we are running it on localhost:5000

Activity:3 Run The app in local browser

```
(base) C:\Users\lenovo\Downloads\z-restaurant recommendation system\z-restaurant recommendation system\flask>python app.
py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 643-633-625
127.0.0.1 - - [24/Nov/2023 14:13:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Nov/2023 14:13:20] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [24/Nov/2023 14:13:27] "POST /recommendation HTTP/1.1" 200 -
```

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page

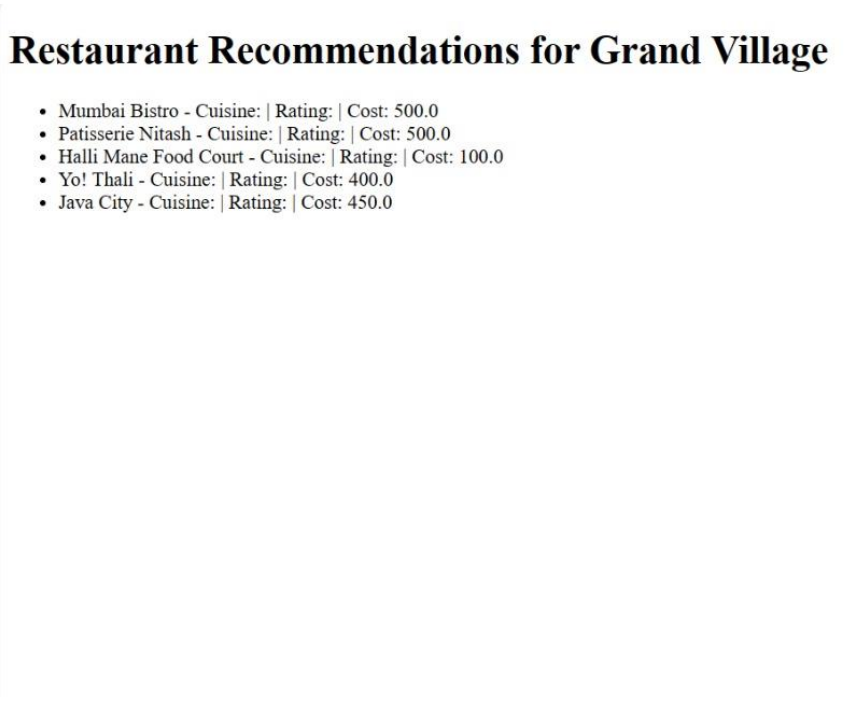
Showcasing The UI



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

I. Checking recommendation for the restaurant: ‘grand village’

This is the prediction page where we will provide a restaurant name for which we will get the top recommended restaurants, which based on cuisines, mean rating (out of 5), cost in thousands.



II. Checking recommendation for the restaurant ‘Cinnamon’

And here is the recommendation

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