RESTAURANT RECOMMENDATION SYSTEM

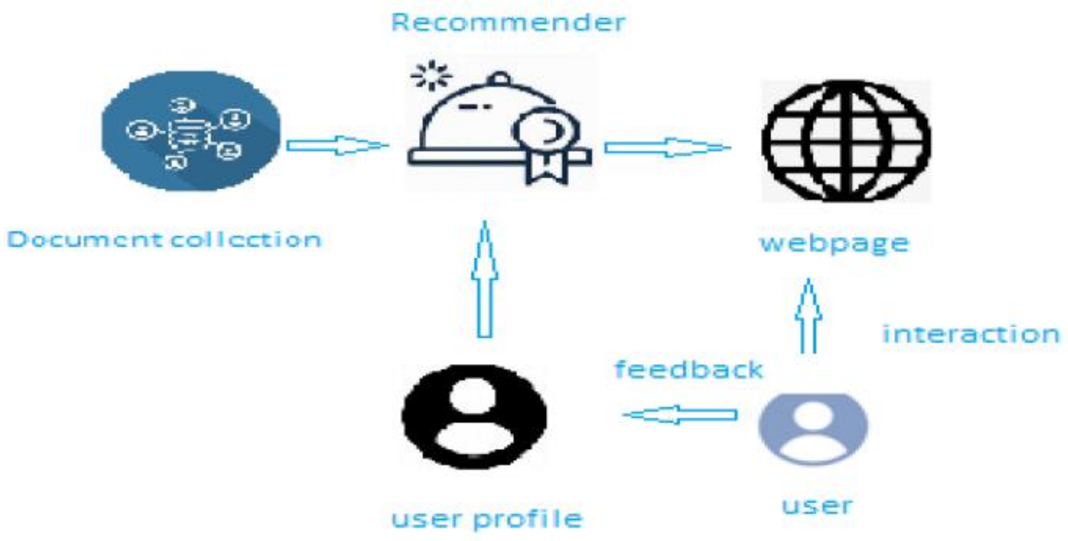
Project Description:

Users of recommendation applications prioritize understanding how they might enjoy a restaurant, particularly when dining with family, friends, or colleagues. Traditionally, individuals relied on suggestions from friends to choose restaurants. While this approach is simple and familiar, it has notable drawbacks. Recommendations from friends are constrained by their own experiences, limiting users to information about places their friends have visited. Additionally, there's a risk that users may not appreciate the places recommended by their friends.

Solution:

We are developing a content-based recommendation system with the objective of providing personalized restaurant suggestions. In this system, users input a restaurant name, and the recommender system analyses reviews from other restaurants. It then recommends additional restaurants with similar reviews, sorted by the highest ratings. This system is particularly beneficial for tourists exploring a new city, as it helps them discover popular restaurants. Additionally, locals can use it to stay updated on newly recommended restaurants based on their preferences and activity within the city.

ARCHITECTURE



Learning Outcomes:

Upon completion of this project, participants will achieve the following learning outcomes:

1. Proficiency in implementing a recommendation system using Content-Based Filtering.
2. Understanding of the data pre-processing phase, including various techniques for cleaning and preparing the data.
3. Ability to analyze and gain insights from data through effective visualization methods.
4. Application of appropriate algorithms based on dataset characteristics and insights derived from visualization.
5. Competence in assessing the accuracy of a model.
6. Mastery of building a web application using the Flask framework.

Pre-requisites:

Prerequisites for successfully completing the project include the installation of the following software and packages:

Activity 1: Install Anaconda IDE / Anaconda Navigator.**

To develop a solution for the given problem statement, an environment is needed for writing and testing code. Anaconda IDE (Integrated Development Environment) is utilized for this purpose.

Activity 2: Install the Following Packages for Building Machine Learning Models**

1. Numpy:

   - Description: An open-source numerical Python library that includes multidimensional array and matrix data structures for performing mathematical operations.

2. Scikit-learn:

- Description: A free machine learning library for Python that supports various algorithms such as support vector machines, random forests, and k-neighbours. It also integrates with Python numerical and scientific libraries like NumPy and SciPy.

3. Matplotlib and Seaborn:

- Matplotlib: Primarily used for basic plotting, offering visualizations like bars, pies, lines, and scatter plots.

- Seaborn: Provides a variety of visualization patterns, using less syntax and featuring easily customizable default themes.

4. Flask:

- Description: A web framework used for building web applications.

Installation Steps using Anaconda Navigator:

1. Open Anaconda Prompt.

2. Type "pip install pandas" and press enter.

3. Type "pip install matplotlib" and press enter.

4. Type "pip install seaborn" and press enter.

5. Type "pip install plotly" and press enter.

6. Type "pip install numpy" and press enter.

7. Type "pip install scikit-image" and press enter.

8. Type "pip install scikit-learn" and press enter.

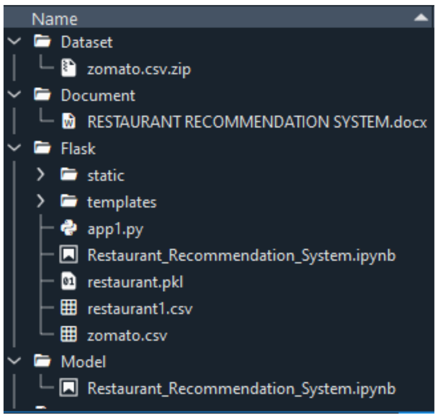
9. Type "pip install Flask" and press enter.

Project Work-Flow

The user engages with the User Interface (UI) to input features.

The integrated model analyzes the entered features.

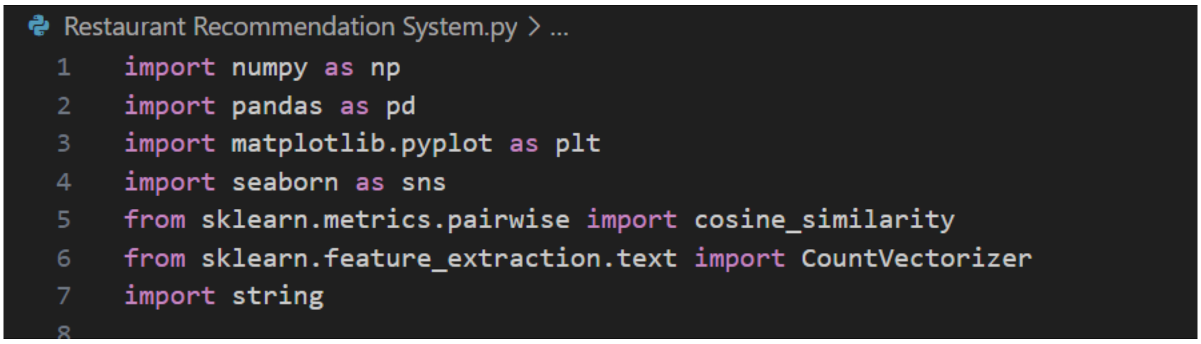
After the model processes the input, the prediction is displayed on the UI.



```
Name
Dataset
  zomato.csv.zip
Document
  RESTAURANT RECOMMENDATION SYSTEM.docx
Flask
  static
  templates
  app1.py
  Restaurant_Recommendation_System.ipynb
  restaurant.pkl
  restaurant1.csv
  zomato.csv
Model
  Restaurant_Recommendation_System.ipynb
```

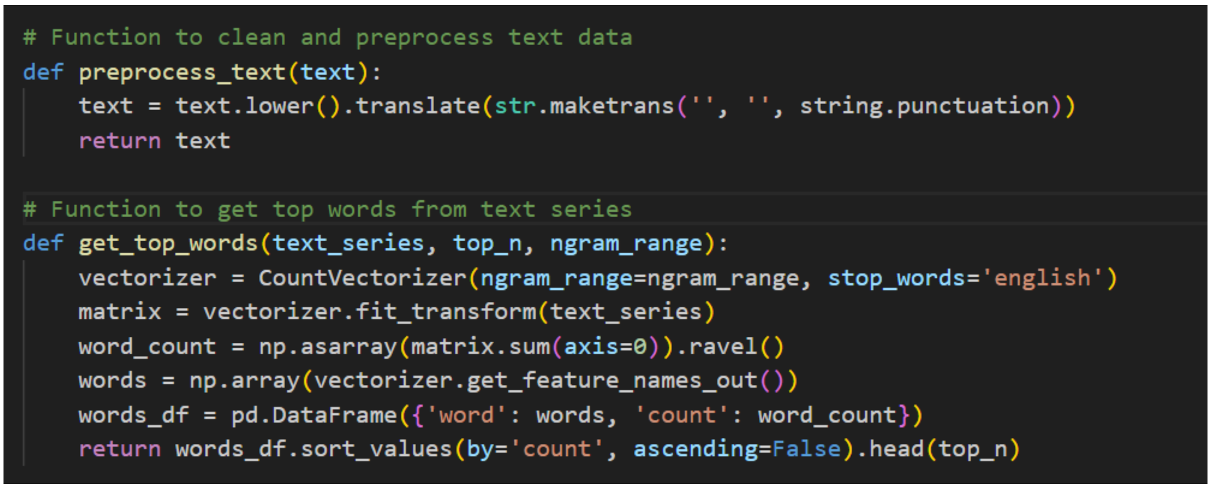
TASKS

1. Importing libraries:



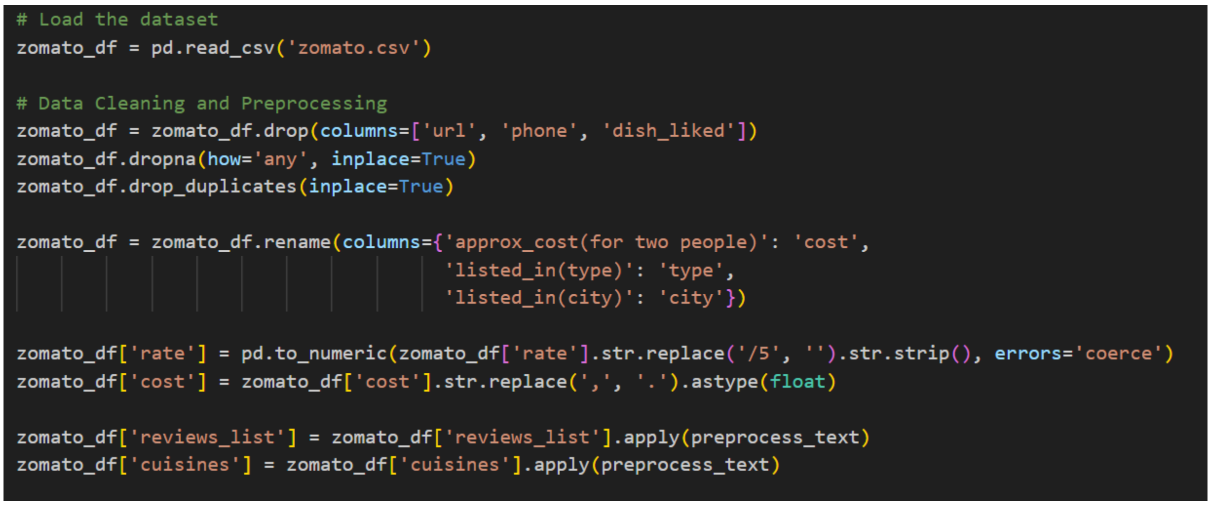
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
import string
```

2. Text Pre-processing functions:



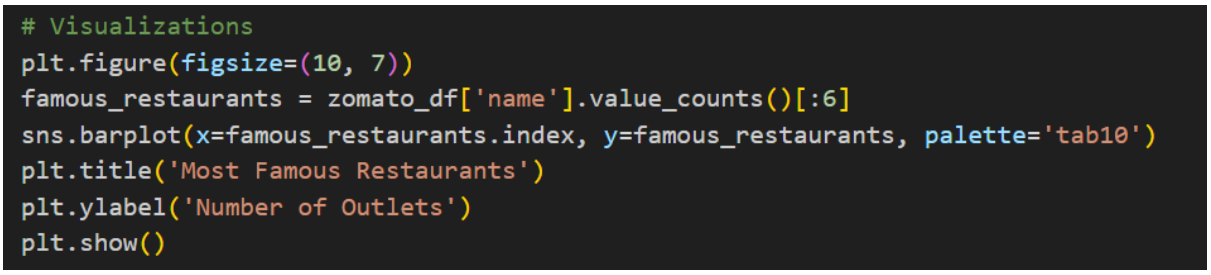
```python
# Function to clean and preprocess text data
def preprocess_text(text):
    text = text.lower().translate(str.maketrans('', '', string.punctuation))
    return text

# Function to get top words from text series
def get_top_words(text_series, top_n, ngram_range):
    vectorizer = CountVectorizer(ngram_range=ngram_range, stop_words='english')
    matrix = vectorizer.fit_transform(text_series)
    word_count = np.asarray(matrix.sum(axis=0)).ravel()
    words = np.array(vectorizer.get_feature_names_out())
    words_df = pd.DataFrame({'word': words, 'count': word_count})
    return words_df.sort_values(by='count', ascending=False).head(top_n)
```

3. Loading Data and Pre-processing

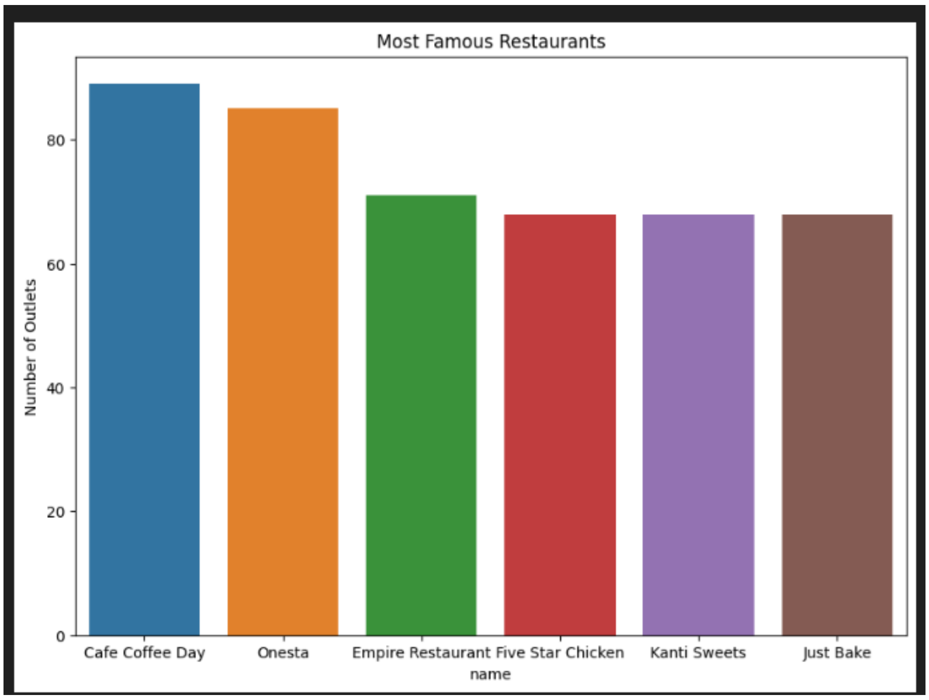


```python
# Load the dataset
zomato_df = pd.read_csv('zomato.csv')

# Data Cleaning and Preprocessing
zomato_df = zomato_df.drop(columns=['url', 'phone', 'dish_liked'])
zomato_df.dropna(how='any', inplace=True)
zomato_df.drop_duplicates(inplace=True)

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

zomato_df['rate'] = pd.to_numeric(zomato_df['rate'].str.replace('/5', '').str.strip(), errors='coerce')
zomato_df['cost'] = zomato_df['cost'].str.replace(',', '.').astype(float)

zomato_df['reviews_list'] = zomato_df['reviews_list'].apply(preprocess_text)
zomato_df['cuisines'] = zomato_df['cuisines'].apply(preprocess_text)
```

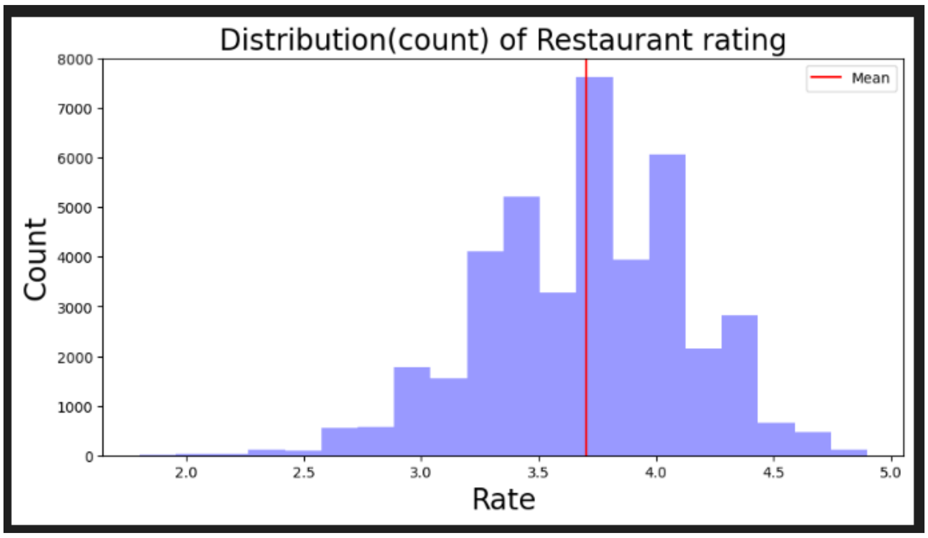
4. Data Visualization:



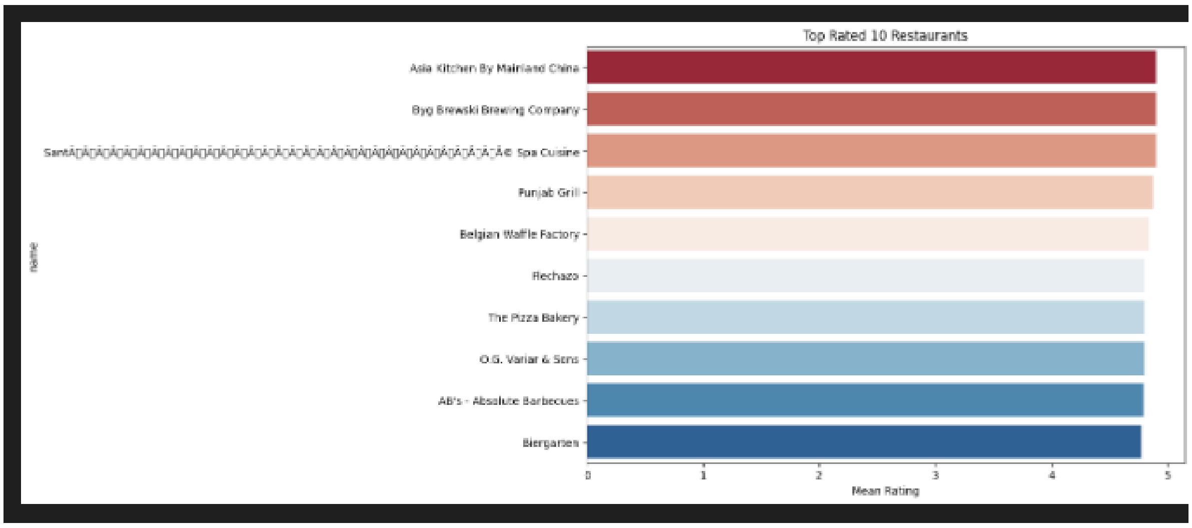
```
# Visualizations
plt.figure(figsize=(10, 7))
famous_restaurants = zomato_df['name'].value_counts()[:6]
sns.barplot(x=famous_restaurants.index, y=famous_restaurants, palette='tab10')
plt.title('Most Famous Restaurants')
plt.ylabel('Number of Outlets')
plt.show()
```



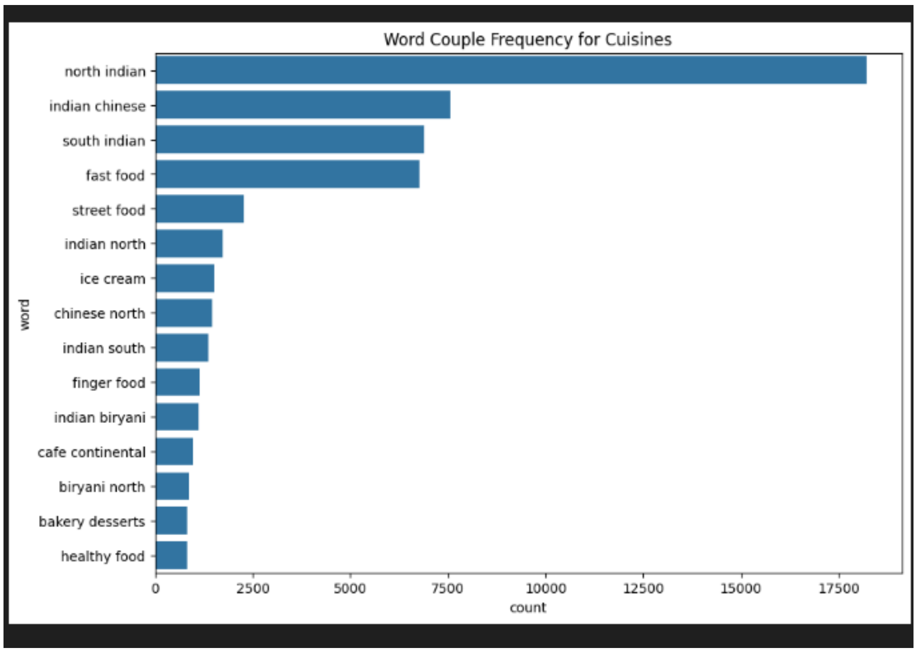
```
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10, 5))
sns.distplot(zomato_df['rate'], kde=False, color='b', ax=ax, bins=20)
ax.axvline(zomato_df['rate'].mean(), 0, 1, color='r', label='Mean')
ax.legend()
ax.set_ylabel('Count', size=20)
ax.set_xlabel('Rate', size=20)
ax.set_title('Distribution(count)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



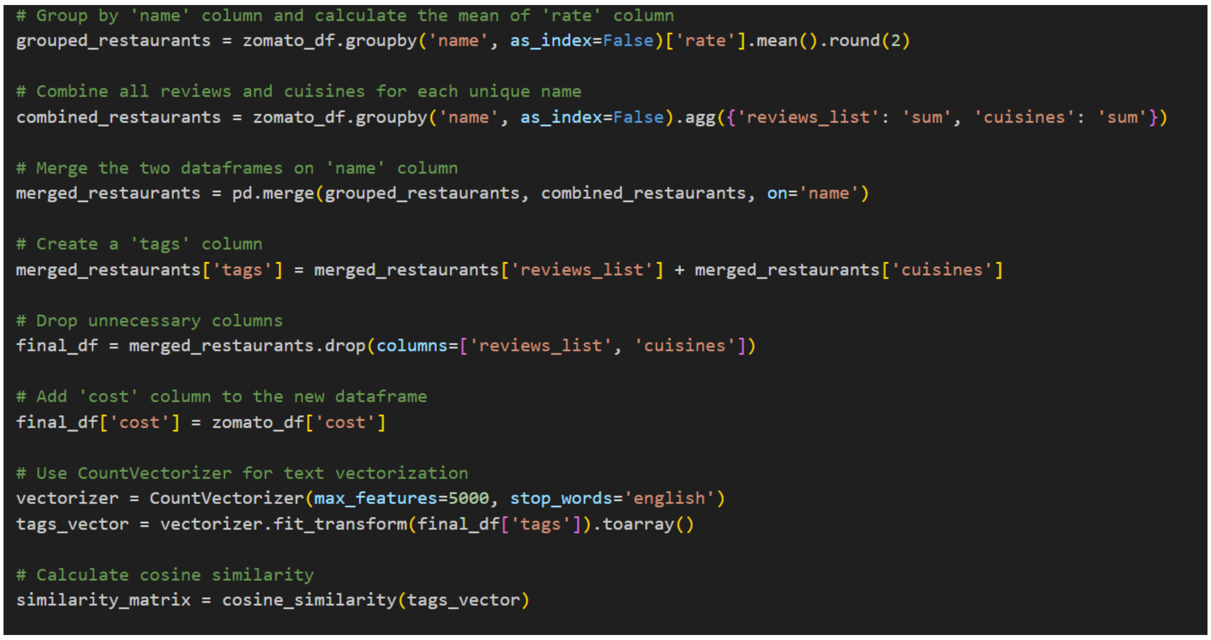
```python
# Top Rated 10 Restaurants Visualization
df_rating = zomato_df.groupby('name', as_index=False)['rate'].mean().round(2)
df_rating = pd.DataFrame({'name': df_rating['name'], 'Mean Rating': df_rating['rate']})
df_rating = df_rating.sort_values(by='Mean Rating', ascending=False).head(10)

plt.figure(figsize=(10, 7))
sns.barplot(data=df_rating, x='Mean Rating', y='name', palette='RdBu')
plt.title('Top Rated 10 Restaurants')
plt.show()
```



```
# Word Couple Frequency for Cuisines Visualization
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')
plt.show()
```

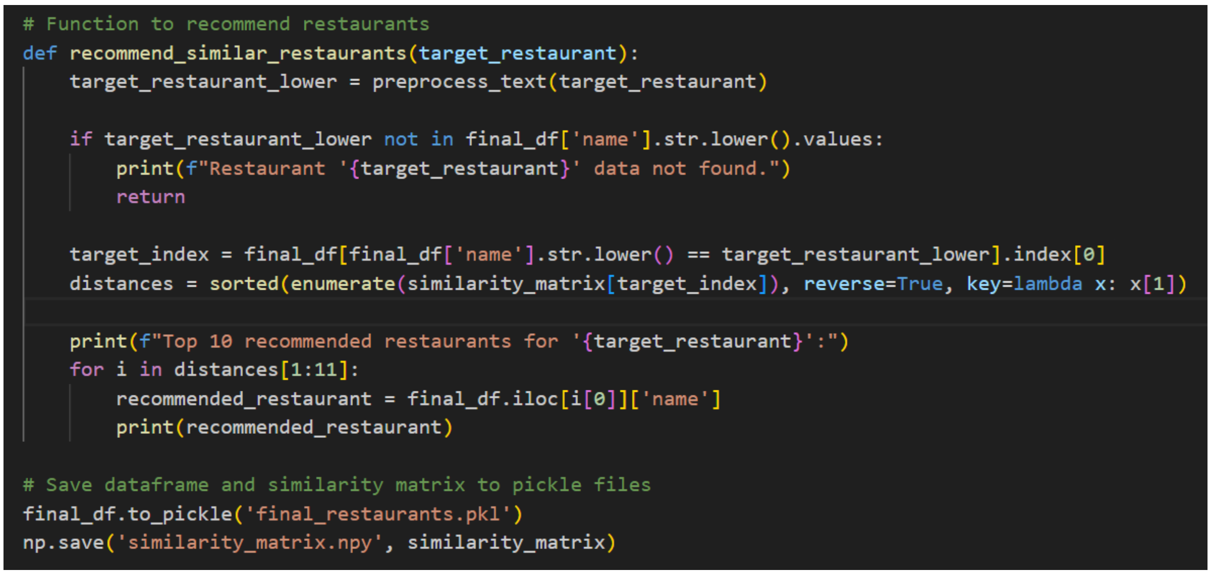


5. Data Transformation and Feature Scaling:



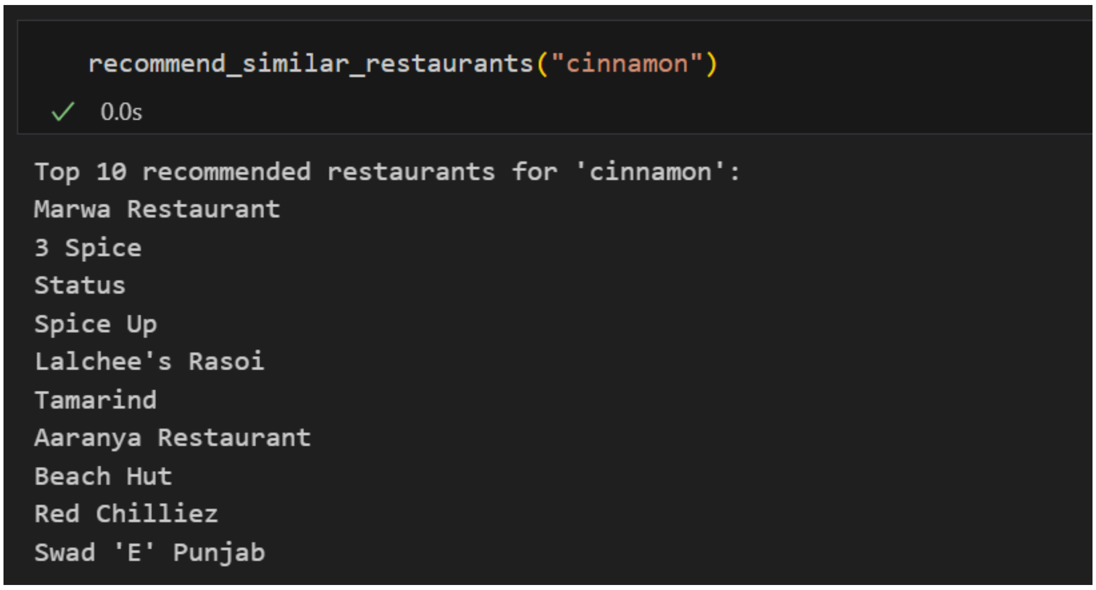
```python
# Group by 'name' column and calculate the mean of 'rate' column
grouped_restaurants = zomato_df.groupby('name', as_index=False)['rate'].mean().round(2)

# Combine all reviews and cuisines for each unique name
combined_restaurants = zomato_df.groupby('name', as_index=False).agg({'reviews_list': 'sum', 'cuisines': 'sum'})

# Merge the two dataframes on 'name' column
merged_restaurants = pd.merge(grouped_restaurants, combined_restaurants, on='name')

# Create a 'tags' column
merged_restaurants['tags'] = merged_restaurants['reviews_list'] + merged_restaurants['cuisines']

# Drop unnecessary columns
final_df = merged_restaurants.drop(columns=['reviews_list', 'cuisines'])

# Add 'cost' column to the new dataframe
final_df['cost'] = zomato_df['cost']

# Use CountVectorizer for text vectorization
vectorizer = CountVectorizer(max_features=5000, stop_words='english')
tags_vector = vectorizer.fit_transform(final_df['tags']).toarray()

# Calculate cosine similarity
similarity_matrix = cosine_similarity(tags_vector)
```

6. Recommendation Function and Saving results:

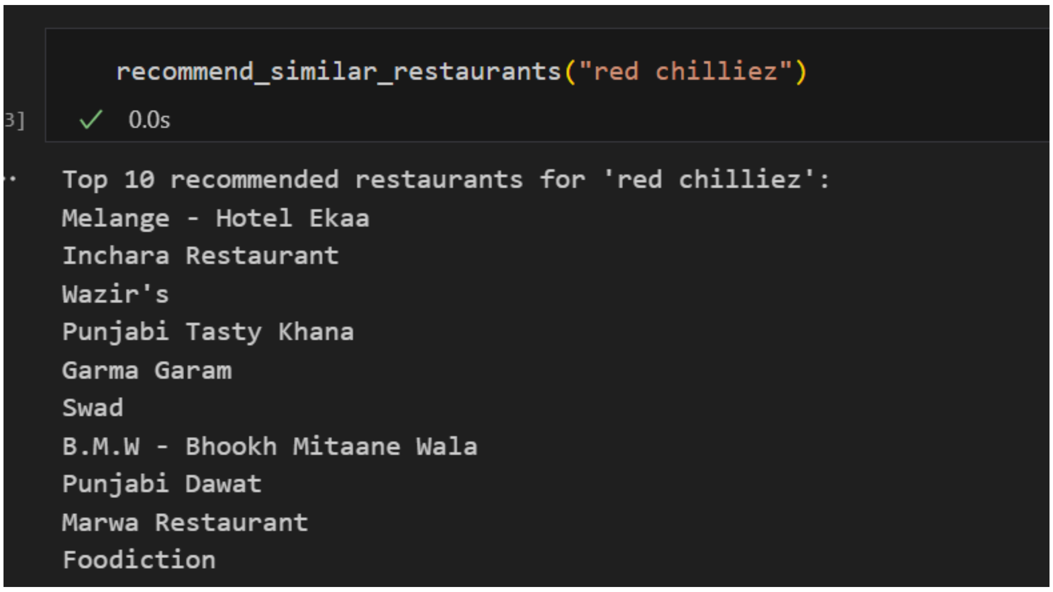


```python
# Function to recommend restaurants
def recommend_similar_restaurants(target_restaurant):
    target_restaurant_lower = preprocess_text(target_restaurant)

    if target_restaurant_lower not in final_df['name'].str.lower().values:
        print(f"Restaurant '{target_restaurant}' data not found.")
        return

    target_index = final_df[final_df['name'].str.lower() == target_restaurant_lower].index[0]
    distances = sorted(enumerate(similarity_matrix[target_index]), reverse=True, key=lambda x: x[1])

    print(f"Top 10 recommended restaurants for '{target_restaurant}':")
    for i in distances[1:11]:
        recommended_restaurant = final_df.iloc[i[0]]['name']
        print(recommended_restaurant)

# Save dataframe and similarity matrix to pickle files
final_df.to_pickle('final_restaurants.pkl')
np.save('similarity_matrix.npy', similarity_matrix)
```

7. Recommendation example:



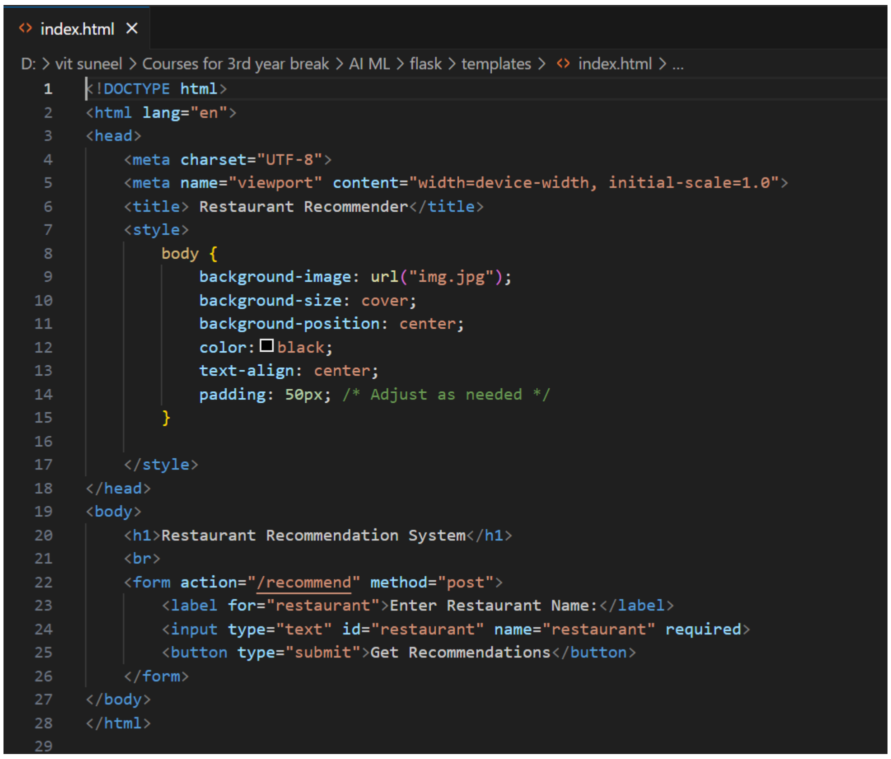
```
recommend_similar_restaurants("cinnamon")
✓ 0.0s

Top 10 recommended restaurants for 'cinnamon':
Marwa Restaurant
3 Spice
Status
Spice Up
Lalchee's Rasoi
Tamarind
Aaranya Restaurant
Beach Hut
Red Chilliez
Swad 'E' Punjab
```



```
recommend_similar_restaurants("red chilliez")
3] ✓ 0.0s

Top 10 recommended restaurants for 'red chilliez':
Melange - Hotel Ekaa
Inchara Restaurant
Wazir's
Punjabi Tasty Khana
Garma Garam
Swad
B.M.W - Bhookh Mitaane Wala
Punjabi Dawat
Marwa Restaurant
Foodiction
```

Application Building:

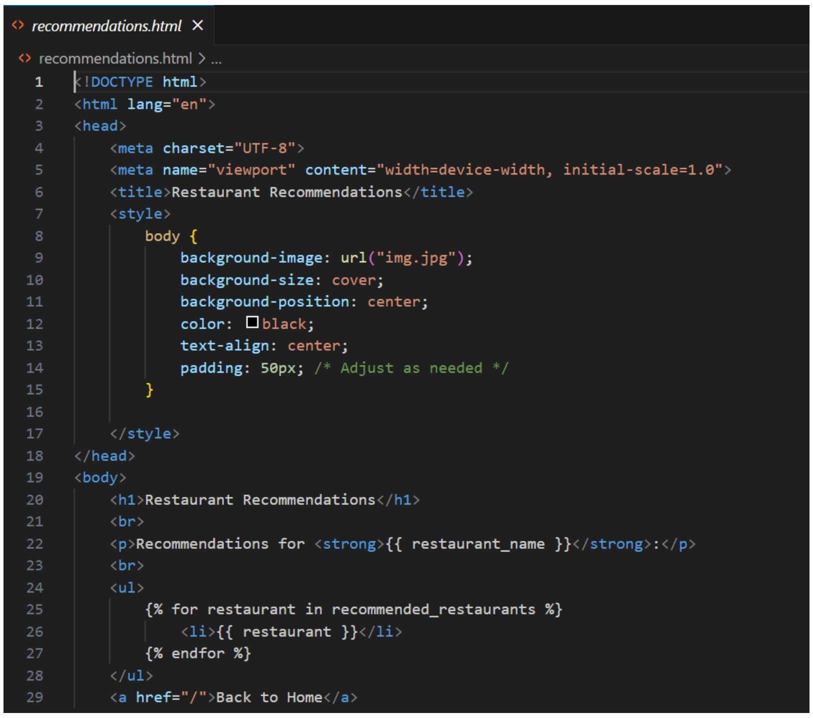
1. Build HTML Files



index.html ✕

D: > vit suneel > Courses for 3rd year break > AI ML > flask > templates > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Restaurant Recommender</title>
    <style>
        body {
            background-image: url("img.jpg");
            background-size: cover;
            background-position: center;
            color: black;
            text-align: center;
            padding: 50px; /* Adjust as needed */
        }

    </style>
</head>
<body>
    <h1>Restaurant Recommendation System</h1>
    <br>
    <form action="/recommend" method="post">
        <label for="restaurant">Enter Restaurant Name:</label>
        <input type="text" id="restaurant" name="restaurant" required>
        <button type="submit">Get Recommendations</button>
    </form>
</body>
</html>
```

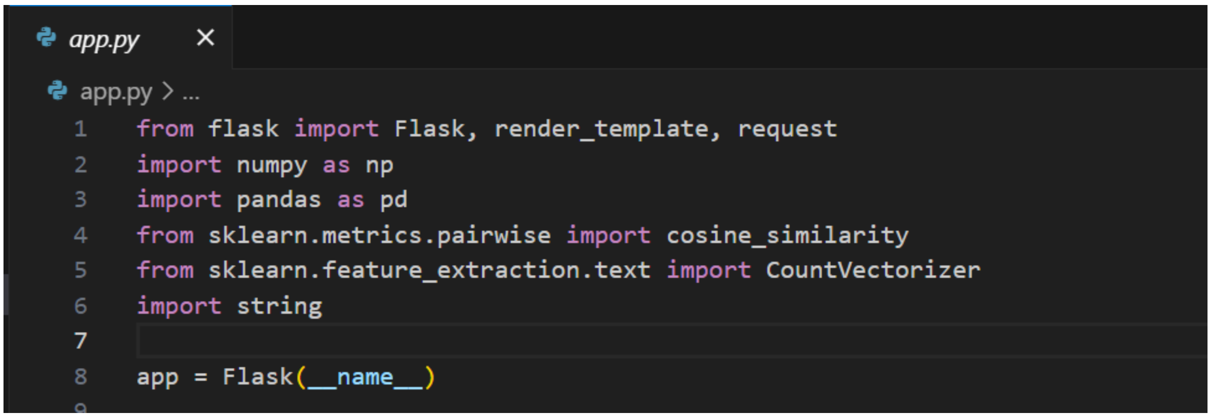


recommendations.html ✕

recommendations.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Restaurant Recommendations</title>
    <style>
        body {
            background-image: url("img.jpg");
            background-size: cover;
            background-position: center;
            color: black;
            text-align: center;
            padding: 50px; /* Adjust as needed */
        }

    </style>
</head>
<body>
    <h1>Restaurant Recommendations</h1>
    <br>
    <p>Recommendations for <strong>{{ restaurant_name }}</strong>:</p>
    <br>
    <ul>
        {% for restaurant in recommended_restaurants %}
            <li>{{ restaurant }}</li>
        {% endfor %}
    </ul>
    <a href="/">Back to Home</a>
```

2. Importing libraries and setting up flask app

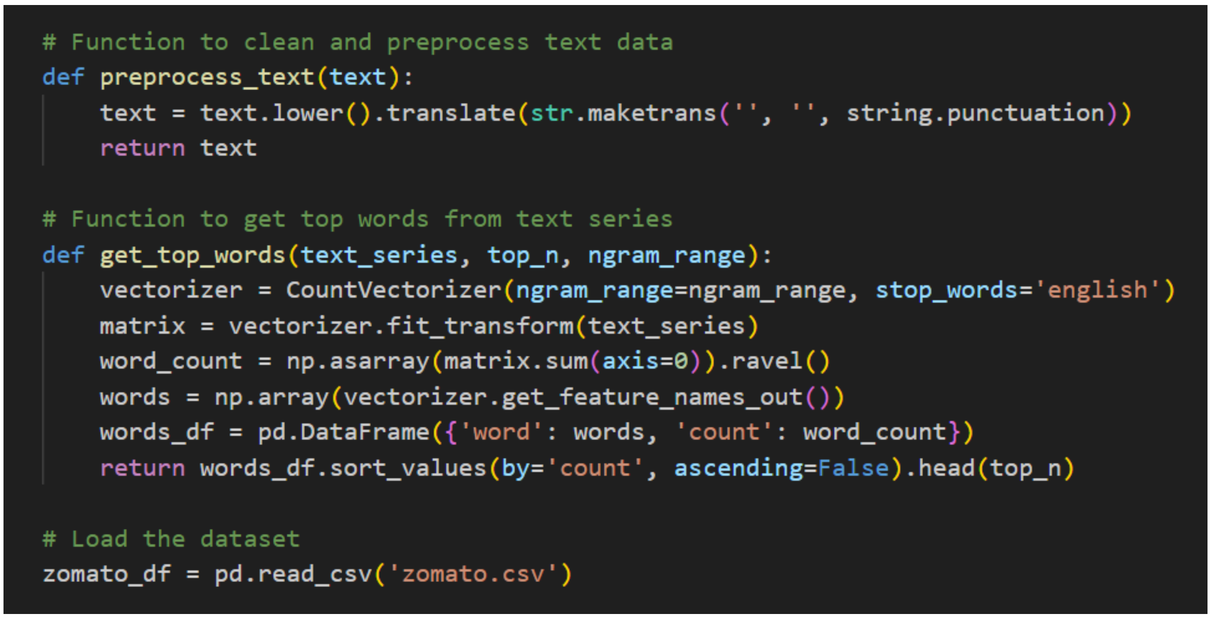


app.py

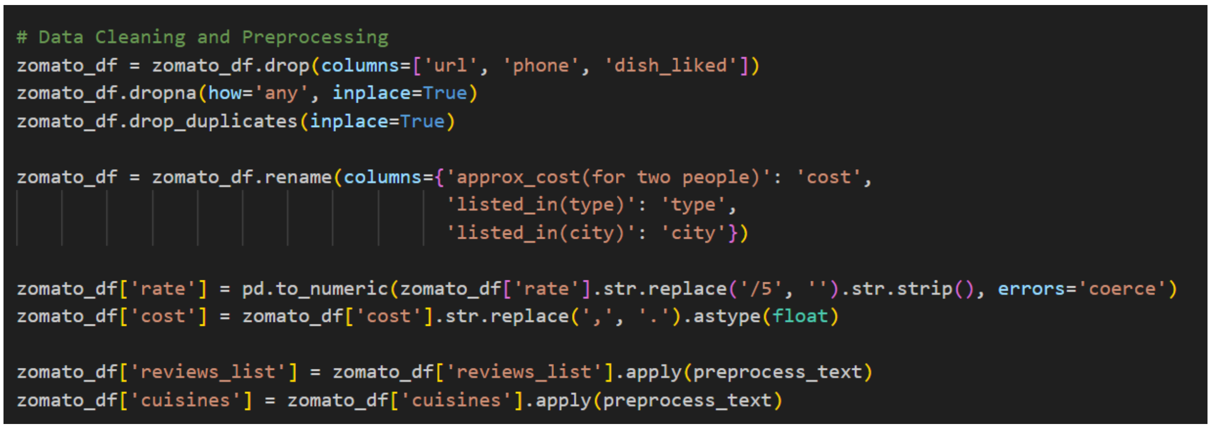
app.py > ...

```python
from flask import Flask, render_template, request
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
import string

app = Flask(__name__)
```

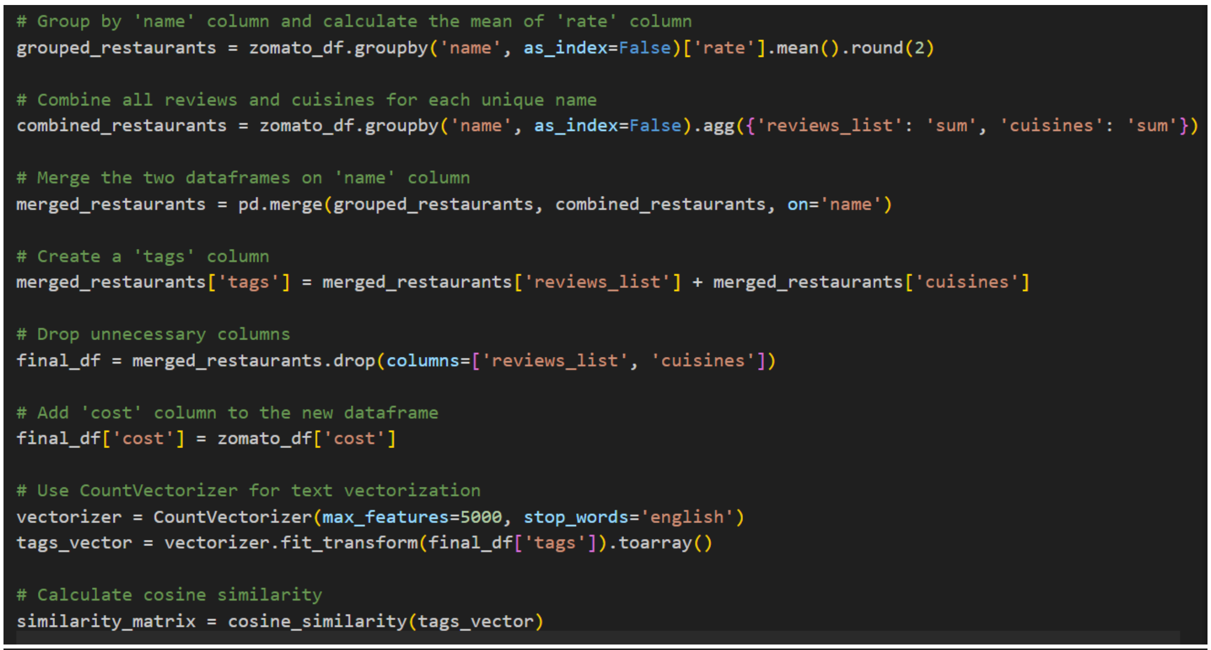
3. Data Pre-processing and recommendation functions



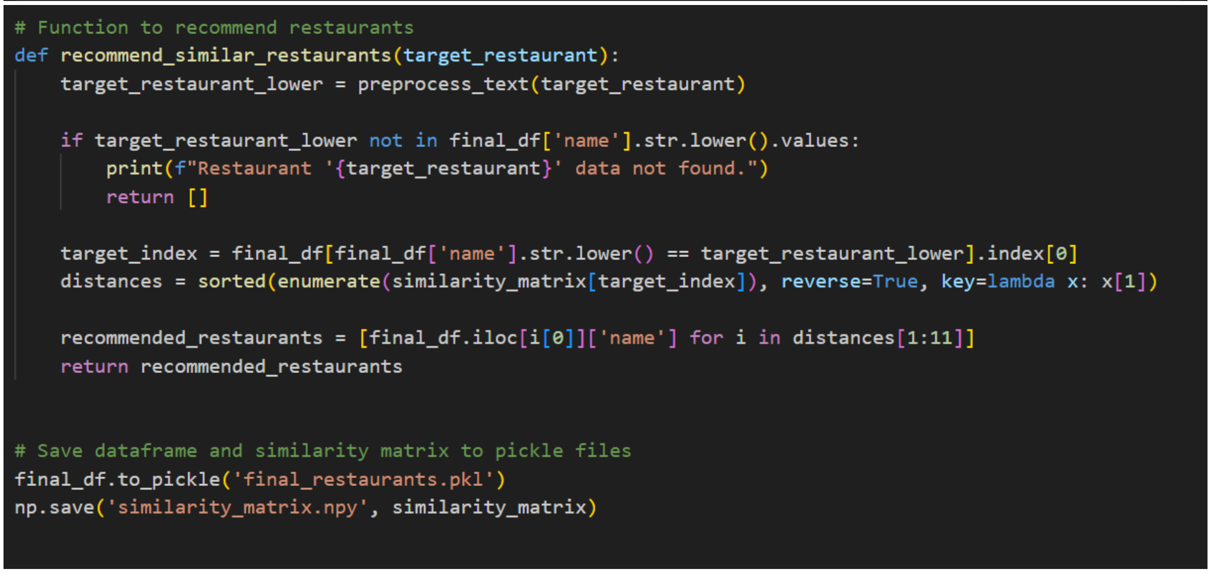
```python
# Function to clean and preprocess text data
def preprocess_text(text):
    text = text.lower().translate(str.maketrans('', '', string.punctuation))
    return text

# Function to get top words from text series
def get_top_words(text_series, top_n, ngram_range):
    vectorizer = CountVectorizer(ngram_range=ngram_range, stop_words='english')
    matrix = vectorizer.fit_transform(text_series)
    word_count = np.asarray(matrix.sum(axis=0)).ravel()
    words = np.array(vectorizer.get_feature_names_out())
    words_df = pd.DataFrame({'word': words, 'count': word_count})
    return words_df.sort_values(by='count', ascending=False).head(top_n)

# Load the dataset
zomato_df = pd.read_csv('zomato.csv')
```



```python
# Data Cleaning and Preprocessing
zomato_df = zomato_df.drop(columns=['url', 'phone', 'dish_liked'])
zomato_df.dropna(how='any', inplace=True)
zomato_df.drop_duplicates(inplace=True)

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

zomato_df['rate'] = pd.to_numeric(zomato_df['rate'].str.replace('/5', '').str.strip(), errors='coerce')
zomato_df['cost'] = zomato_df['cost'].str.replace(',', '.').astype(float)

zomato_df['reviews_list'] = zomato_df['reviews_list'].apply(preprocess_text)
zomato_df['cuisines'] = zomato_df['cuisines'].apply(preprocess_text)
```

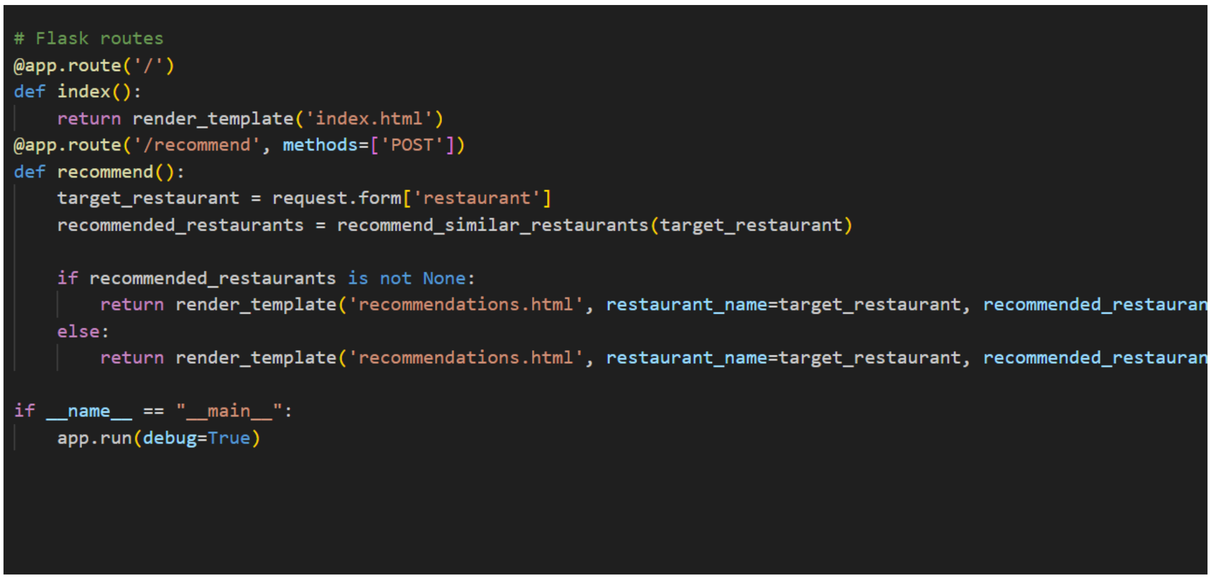


```python
# Group by 'name' column and calculate the mean of 'rate' column
grouped_restaurants = zomato_df.groupby('name', as_index=False)['rate'].mean().round(2)

# Combine all reviews and cuisines for each unique name
combined_restaurants = zomato_df.groupby('name', as_index=False).agg({'reviews_list': 'sum', 'cuisines': 'sum'})

# Merge the two dataframes on 'name' column
merged_restaurants = pd.merge(grouped_restaurants, combined_restaurants, on='name')

# Create a 'tags' column
merged_restaurants['tags'] = merged_restaurants['reviews_list'] + merged_restaurants['cuisines']

# Drop unnecessary columns
final_df = merged_restaurants.drop(columns=['reviews_list', 'cuisines'])

# Add 'cost' column to the new dataframe
final_df['cost'] = zomato_df['cost']

# Use CountVectorizer for text vectorization
vectorizer = CountVectorizer(max_features=5000, stop_words='english')
tags_vector = vectorizer.fit_transform(final_df['tags']).toarray()

# Calculate cosine similarity
similarity_matrix = cosine_similarity(tags_vector)
```



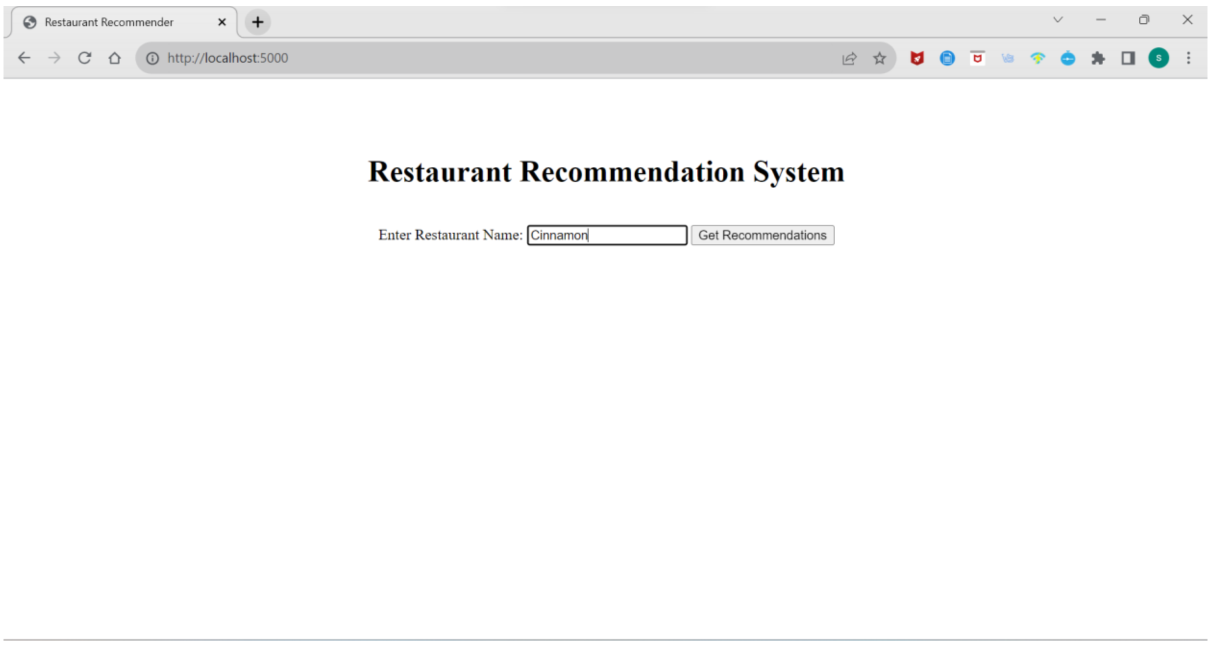
```python
# Function to recommend restaurants
def recommend_similar_restaurants(target_restaurant):
    target_restaurant_lower = preprocess_text(target_restaurant)

    if target_restaurant_lower not in final_df['name'].str.lower().values:
        print(f"Restaurant '{target_restaurant}' data not found.")
        return []

    target_index = final_df[final_df['name'].str.lower() == target_restaurant_lower].index[0]
    distances = sorted(enumerate(similarity_matrix[target_index]), reverse=True, key=lambda x: x[1])

    recommended_restaurants = [final_df.iloc[i[0]]['name'] for i in distances[1:11]]
    return recommended_restaurants


# Save dataframe and similarity matrix to pickle files
final_df.to_pickle('final_restaurants.pkl')
np.save('similarity_matrix.npy', similarity_matrix)
```

4. Flask Routes



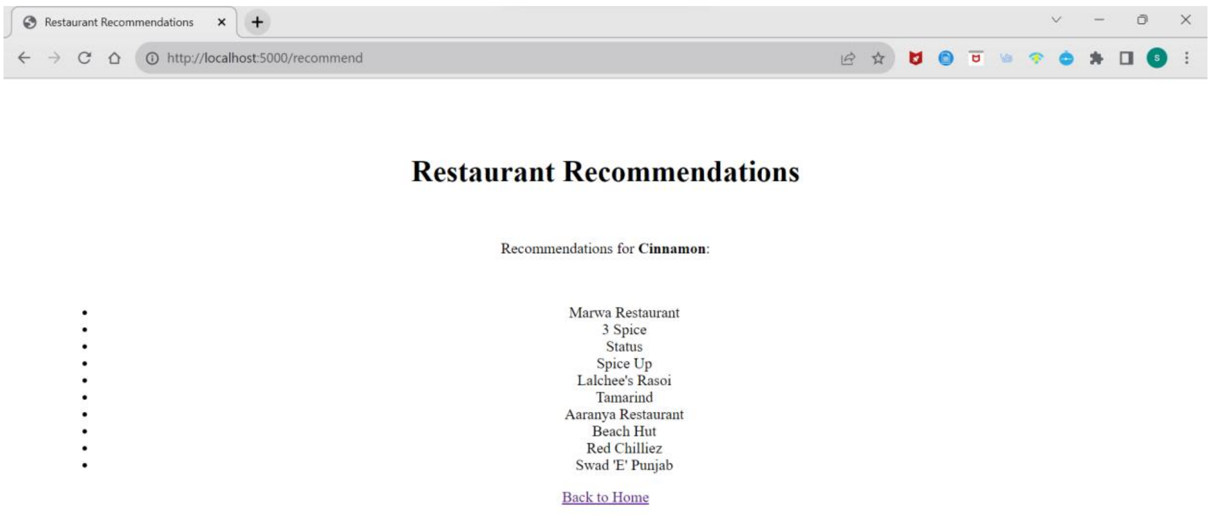
```
# Flask routes
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/recommend', methods=['POST'])
def recommend():
    target_restaurant = request.form['restaurant']
    recommended_restaurants = recommend_similar_restaurants(target_restaurant)

    if recommended_restaurants is not None:
        return render_template('recommendations.html', restaurant_name=target_restaurant, recommended_restauran
    else:
        return render_template('recommendations.html', restaurant_name=target_restaurant, recommended_restauran

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

Run App in Local Browser



Restaurant Recommendation System

Enter Restaurant Name: Cinnamon



Restaurant Recommendations

Recommendations for **Cinnamon**:

- Marwa Restaurant
- 3 Spice
- Status
- Spice Up
- Lalchee's Rasoi
- Tamarind
- Aaranya Restaurant
- Beach Hut
- Red Chilliez
- Swad 'E' Punjab

Back to Home