**Restaurant Recommendation System**

**Campus- VIT-AP**

By

V. S Sai Vamsi Veshesh-20BCE7476

Hem Chalasani -20BCD7075

S. Srikar-20BCE7052

**1. INTRODUCTION**

* 1. **Overview:**

The restaurant recommendation system project aims to develop a user-friendly recommendation system that assists users in finding suitable restaurants based on their preferences and constraints. Although reviews give precious recommendations about eateries, reading reviews for businesses can be time consuming. People go through several paragraphs of recommendations published on some websites just wanting to find a right eatery, which has most possible way to satisfy their particular appetite or their own food preference. Above that, when people visit a new place, move or trip, people must go through the same hunt procedure again. This can be tedious and eventually annoying. By leveraging data analysis techniques and machine learning algorithms, the system generates personalized recommendations tailored to individual tastes and requirements. The project encompasses key components such as user interface design, data collection and preprocessing, feature engineering, recommendation model development, and system evaluation. The anticipated outcomes of the project include an efficient and functional restaurant recommendation system, improved user experience and satisfaction, increased efficiency in decision-making processes, and evaluation results indicating the system's performance and effectiveness. Overall, the project aims to provide users with accurate and personalized restaurant recommendations, enhancing their dining experiences and simplifying the process of selecting suitable dining options.

* 1. **Purpose:**

The primary purpose of the restaurant recommendation system is to simplify the process of finding appropriate dining options for users, saving them time and effort in searching for restaurants manually. For an instance, In the case of choosing a restaurant to have lunch when you are traveling to other countries or metropolises, generally you'll ask your friend that lives in that country or megacity what's the best eatery in city. The problem is you don’t have any friends that lived in that city. Your recommendations can be generated by an artificial friend the recommender system. This is the underpinning purpose of the system to help the people in picking the good eateries. By offering personalized recommendations, the system aims to enhance user satisfaction and dining experiences, increasing the likelihood of users discovering new and enjoyable dining venues. Additionally, the project aims to leverage user feedback and ratings to continuously improve the recommendation system's performance and accuracy. The system's purpose is to provide relevant and reliable recommendations that align with the user's preferences, taking into account factors such as restaurant ratings, cuisine types, proximity, and user reviews. Ultimately, the restaurant recommendation system project's purpose is to streamline the restaurant selection process, enhance user satisfaction, and simplify decision-making for users, resulting in an improved dining experience.

**2. Literature Survey**

**2.1 Existing problem:**

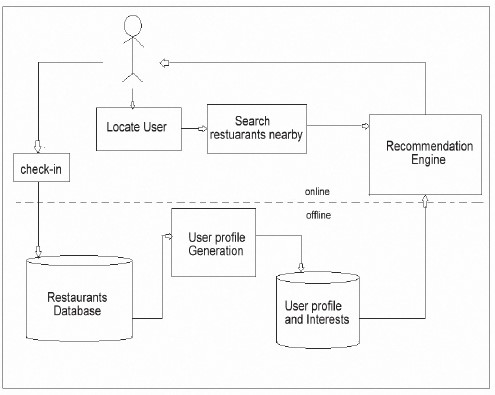
While multiple being recommender systems substantially target individualities, there's a remarkable increase of recommender systems which induce suggestions for groups. Some early systems were developed in a variety of disciplines, similar as group web page recommendation (Lieberman et al. 1999), tour packages for groups of tourists (Ardissono et al. 2003), music tracks and playlists for large groups of many listeners (Crossen et al. 2002), movies and TV programs for friends and family (O’Connor et al. 2001; Yu et al. 2006). Group scripts are especially popular in the food sphere in which a group of family members, friends or associates wants to throw a party or simply have a mess together. still, the complexity significantly increases when food recommender systems need to take into account the preferences of all group members and strategies for achieving the agreement within group members. From the check, we've inferred that they've developed a recommended system just to search for food. still, the complexity significantly increases when food recommender systems need to take into account the preferences of all group members and strategies for achieving the agreement within group members. numerous caffs store and maintain their day to day deals manually. But some of them are having robotization system which is helping them to store the data. But similar caffs are storing the information about the orders and the client information. They don’t have installation to store the information of feedbacks and favourite orders of guests over some period of time. caffs are having standalone operations so at one time, they've the installation of numerous defences or numerous operations which is passing at one time. So, they're storing them and also at last, the eatery directors will be suitable to see the data of last day.

**2.2 Proposed solution:**

We're putting forth a method that makes use of Stream lit to create a quick web application. The creation of a system that recommends restaurants to users based on their preferences is our main goal. since each person consumes a different type of food to make the most of the data represented in user reviews, we carefully choose features based on tastes and dietary constraints. We use Latent Factor Collaborative Filtering Optimisation to create a restaurant recommendation system. According to a user's or group of users' tastes, such as a lovely ambiance, good meals, tasty desserts, and so forth, this system suggests eateries. Users who use our technology receive tailored restaurant recommendations. The user's location is taken into account when making a recommendation for a new user, as is the feedback we received.

**3. THEORITICAL ANALYSIS**

**3.1 Block diagram**



**3.2 Hardware / Software designing**

**Hardware Requirements**

* NVIDIA GPU
* Storage
* Python and TensorFlow or PyTorch

**Software Requirements**

* VS Code
* Flask
* Python

**4. EXPERIMENTAL INVESTIGATIONS**

Recommendation systems are a type of information filtering systems because they improve the quality of search results and provide elements that are more relevant to the search item or that are related to the search history of the user.

These are active information filtering systems that personalize the information provided to a user based on their interests, relevance of the information, etc. Recommendation systems are widely used to recommend movies, items, restaurants, places to visit, items to buy, etc.

There are two types of recommendation systems:

**1. Content Based Recommender System**

This type of recommender system is useful for solving cold start problems. Here you use the past spending habits of a user and their past order details along with the past record of a vendor to track what rating they are most likely to give to that particular order this ultimately turns out to be one of our very familiar datasets type that is Regression. Here we can use our vendor rating per order as a target and rest other data as input and train a Regression model that helps us to predict what ratings user is going to give and based on that we can recommend those restaurants which tend to get a good rating or above 4 ratings.

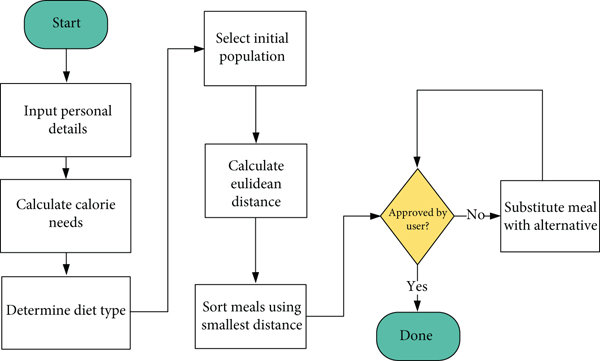
**2. Collaborative Filtering**

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.

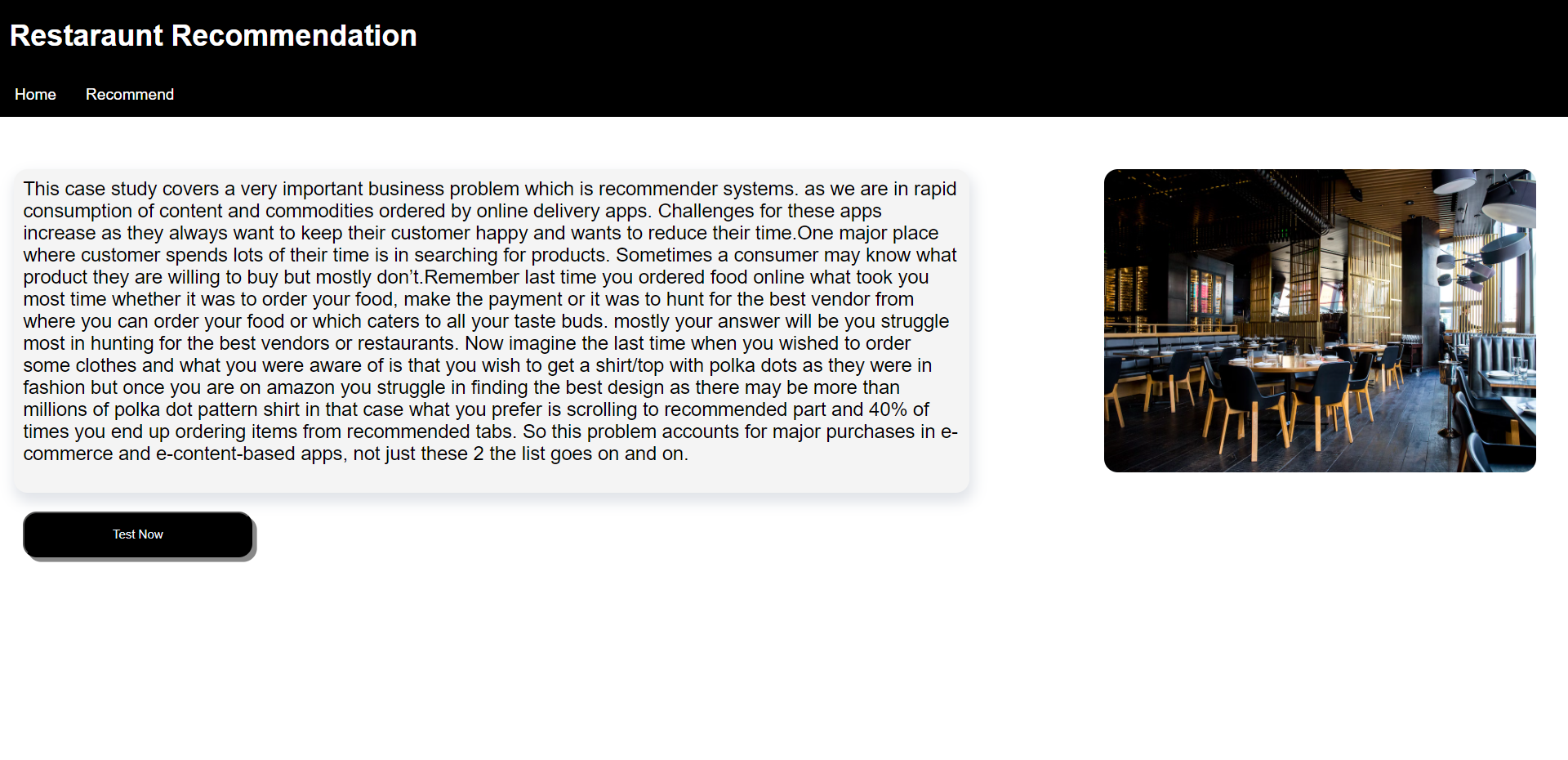
It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions.

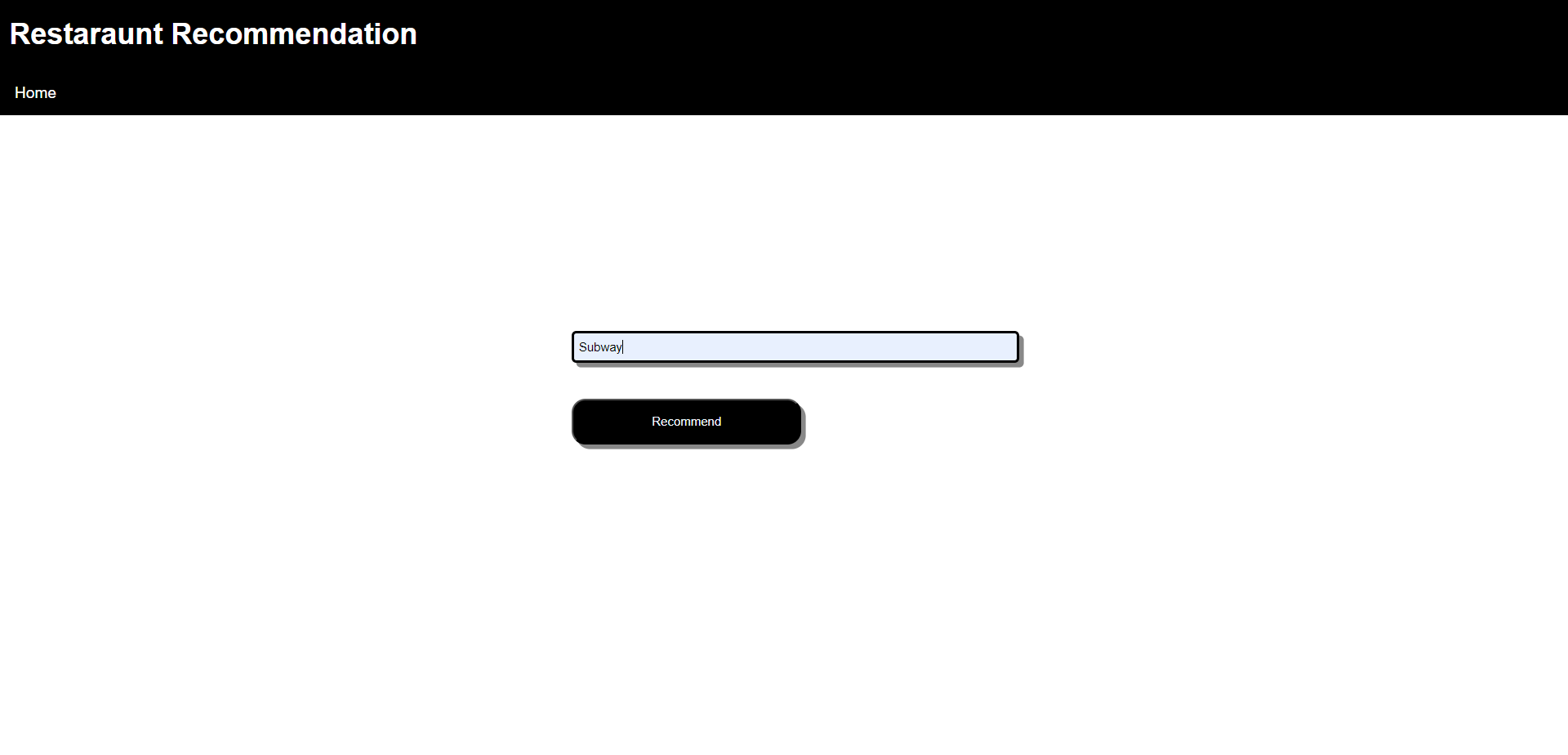
There are many ways to decide which users are similar and combine their choices to create a list of recommendations. This article will show you how to do that with Python. This can be understood as similarities between 2 things whether it may be users or items. As they are just similarities, they are not dependent on understanding data completely but they also arise with a problem of cold start that is whenever you are having a new user this method fails as you have no rating data about users.

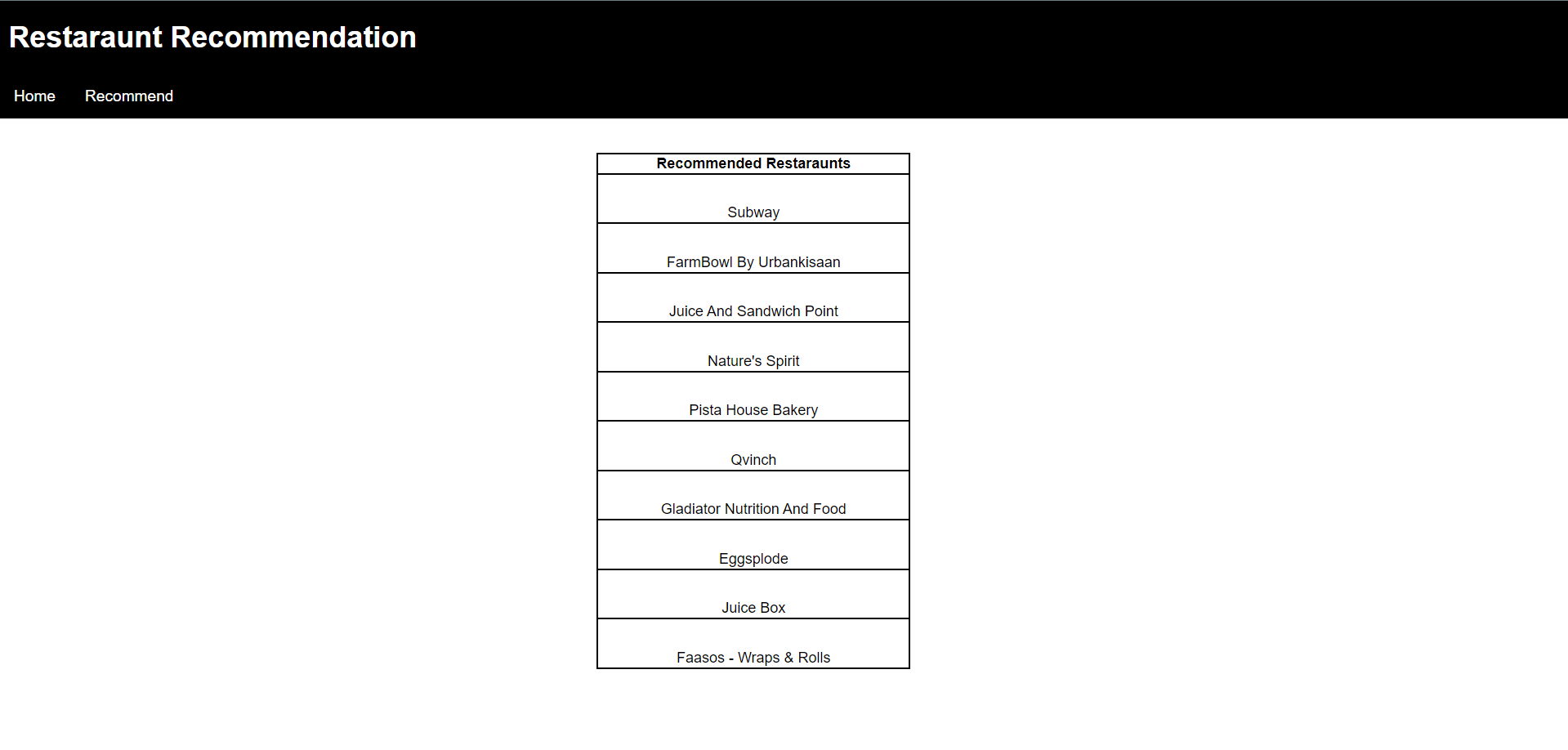
**5. FLOWCHART**



**6. RESULT**

****

****

****

**7.ADVANTAGES & DISADVANTAGES**

**Advantages:**

1. Personalized Recommendations: A restaurant recommendation system can offer personalized suggestions based on the user's preferences, previous choices, and behavior. This enhances the user experience by providing tailored recommendations that match their tastes and dietary requirements.

2. Time and Effort Saving: Users no longer need to manually search for restaurants, read reviews, and compare options. The recommendation system streamlines the process by presenting them with relevant choices, saving time and effort in decision-making.

3. Improved Customer Satisfaction: By suggesting restaurants that align with the user's preferences, the recommendation system increases the likelihood of a positive dining experience. Users are more likely to be satisfied with their choices, leading to increased customer loyalty and positive reviews.

4. Discovery of New Options: Restaurant recommendation systems can introduce users to new and lesser-known establishments that they may not have discovered otherwise. This promotes diversity in dining experiences and supports local businesses by generating exposure for hidden gems.

5. Enhanced User Engagement: A recommendation system can engage users by providing interactive features such as ratings, reviews, and social sharing options. This fosters a sense of community and encourages users to contribute their opinions, leading to a more dynamic and vibrant platform.

Disadvantages:

1. Limited User Input: Recommendation systems heavily rely on user data to provide accurate recommendations. If users have limited input or fail to provide accurate information about their preferences, the system may struggle to deliver relevant suggestions.

2. Lack of Serendipity: Restaurant recommendation systems tend to focus on providing options that closely align with the user's preferences. While this improves accuracy, it may also limit serendipitous discoveries and the exploration of diverse culinary experiences.

3. Bias and Homogeneity: Recommendation algorithms can inadvertently reinforce existing biases by suggesting popular restaurants or favoring certain cuisines. This may lead to a lack of representation for smaller establishments or underrepresented culinary traditions.

4. Overreliance on Ratings: Recommendation systems often prioritize highly rated restaurants, assuming they are superior choices. However, this approach may not consider individual preferences or account for niche establishments that cater to specific tastes.

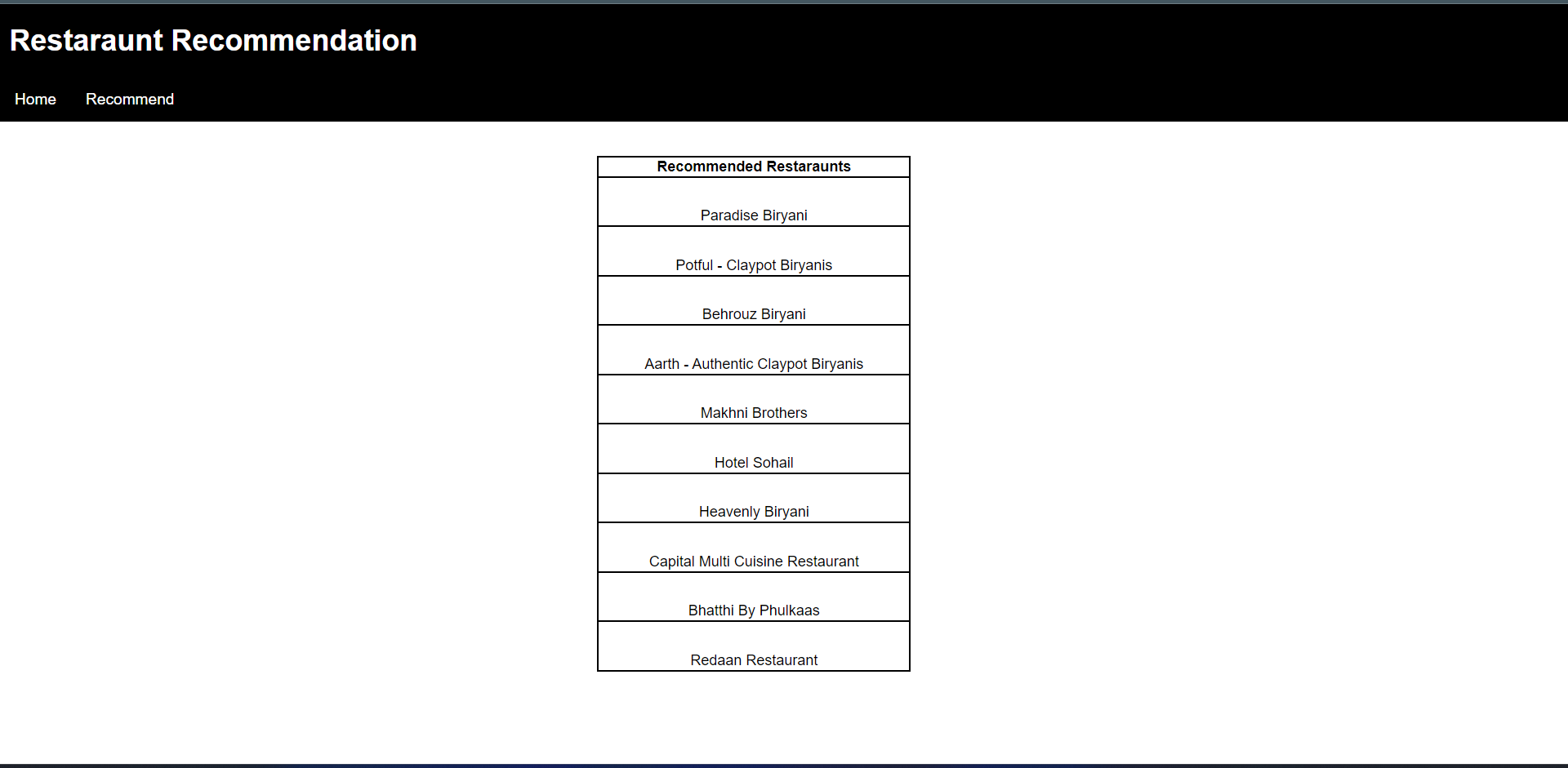
5. Privacy Concerns: To offer personalized recommendations, restaurant recommendation systems collect and analyze user data. This raises privacy concerns, especially if the data is not handled securely or transparently. Users may be wary of sharing personal information, affecting system performance.

**8. APPLICATIONS**

Eat-Smart allows user to search, view and rate restaurants, while collecting user preferences and experiences to perform personalized recommendations. The primary feature is the search function to find restaurant and food options in a specific location. User can review and bookmark businesses and the data will be collected by the application. Once sufficient data is obtained from the user, recommendations are generated using a recommendation model. The following sections demonstrates the features of all pages in the web application.

**9. CONCLUSION**

A restaurant recommender system is an application that recommends similar restaurants to a customer according to the customer’s taste. I hope you liked this article on building a restaurant recommender system using Python. Feel free to ask valuable questions in the comments section below

****

**10. FUTURE SCOPE**

The restaurant recommendation system project lays the foundation for potential future advancements and expansions. Here are some key areas of future scope for the project:

1. Enhanced Recommendation Algorithms: The project can explore more advanced recommendation algorithms, such as hybrid approaches that combine collaborative filtering and content-based filtering techniques. Incorporating deep learning models or reinforcement learning techniques can further improve the accuracy and personalization of recommendations.

2. Integration with External APIs: The system can be extended to integrate with external APIs, such as social media platforms or restaurant reservation systems. This integration can provide real-time updates on restaurant availability, specials, or promotions, enhancing the relevance and timeliness of recommendations.

3. Intelligent Filtering Options: Future iterations of the system can include intelligent filtering options, allowing users to customize recommendation results based on specific criteria. This can include filtering by specific cuisines, price ranges, dietary restrictions, or specific features like outdoor seating or vegetarian-friendly options.

4. Location-Based Services: Leveraging location-based services can provide users with recommendations based on their current location or preferences for specific areas. Integration with mapping services can offer navigation and directions to recommended restaurants, enhancing user convenience.

5. Personalized Profiles: The system can introduce personalized user profiles where users can save their preferences, dietary restrictions, and favorite restaurants. These profiles can be used to deliver more accurate and tailored recommendations based on individual preferences and past dining experiences.

6. Sentiment Analysis: Implementing sentiment analysis techniques can allow the system to analyze user reviews and feedback to identify positive and negative sentiments. This analysis can further improve the recommendation system by considering sentiment scores and incorporating user sentiment preferences into the recommendation process.

7. Integration with Reservation Systems: Integrating the recommendation system with restaurant reservation systems can provide users with the ability to make reservations directly through the platform. This seamless integration can streamline the dining experience for users.

8. Multi-platform Access: Expanding the project to support multiple platforms, such as mobile applications or voice assistants, can enhance user accessibility and convenience.

Overall, the future scope of the restaurant recommendation system project lies in refining and expanding its capabilities to deliver more accurate, personalized, and convenient restaurant recommendations. By incorporating advanced algorithms, intelligent filtering options, and integration with external services, the system can further enhance user satisfaction and provide a seamless dining experience.

**11. BIBILOGRAPHY**

* [**https://www.kaggle.com/datasets/deewakarchakraborty/zomato-restaurant-dataset?resource=download**](https://www.kaggle.com/datasets/deewakarchakraborty/zomato-restaurant-dataset?resource=download)
* [**https://www.analyticsvidhya.com/blog/2022/06/all-about-restaurant-recommender/**](https://www.analyticsvidhya.com/blog/2022/06/all-about-restaurant-recommender/)
* [**https://towardsdatascience.com/how-to-build-a-restaurant-recommendation-system-using-latent-factor-collaborative-filtering-ffe08dd57dca**](https://towardsdatascience.com/how-to-build-a-restaurant-recommendation-system-using-latent-factor-collaborative-filtering-ffe08dd57dca)

**APPENDIX**

1. **Source Code**

from flask import Flask, jsonify,render\_template,url\_for

**Python**

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from tqdm.notebook import tqdm

from sklearn.decomposition import PCA

from sklearn.manifold import TSNE

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.metrics import euclidean\_distances

from scipy.spatial.distance import cdist

from sklearn.feature\_extraction import text

from sklearn.metrics.pairwise import cosine\_similarity

import warnings

warnings.filterwarnings("ignore")

warnings.filterwarnings("always")

from flask import Flask,render\_template,url\_for

from flask import request

# In[95]:

data=pd.read\_csv("HyderabadResturants.csv")

data.drop('links', axis=1, inplace = True)

data.head()

app=Flask(\_\_name\_\_)

# In[96]:

data.info()

# In[97]:

data.shape

# In[98]:

data[["cuisine1", "cuisine2",'cuisine3','cuisine4' ,'cuisine5','cuisine6','cuisine7','cuisine8']] = (

data["cuisine"].str.split(",", expand=True)

)

data.rename(columns ={'price for one': 'price'},inplace=True)

data.head()

# In[99]:

features=["cuisine1","cuisine2","cuisine3","cuisine4","cuisine5","cuisine6","cuisine7","cuisine8"]

data["temp"]=data[features].isnull().sum(axis=1)

data["no\_Of\_cusines"]=8-data["temp"]

data.head()

# In[100]:

feature = data["cuisine"].tolist()

tfidf = text.TfidfVectorizer()

tfidf\_matrix = tfidf.fit\_transform(feature)

similarity = cosine\_similarity(tfidf\_matrix)

# In[101]:

indices = pd.Series(data.index, index=data['names']).drop\_duplicates()

# In[102]:

def restaurant\_recommendation(name, similarity = similarity):

index = indices[name]

similarity\_scores = list(enumerate(similarity[index]))

similarity\_scores = sorted(similarity\_scores, key=lambda x: x[1], reverse=True)

similarity\_scores = similarity\_scores[0:10]

restaurantindices = [i[0] for i in similarity\_scores]

return data['names'].iloc[restaurantindices]

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/search')

def search():

return render\_template('search.html')

@app.route('/index', methods=['POST'])

def keywords():

output=request.form.get('output')

return render\_template( "res.html", rest=list(restaurant\_recommendation(output).to\_dict().values()) )

@app.route('/res',methods=['POST'])

def res():

data=request.form.get('output')

return render\_template("res.html",rest=data)

if \_\_name\_\_=='\_\_main\_\_':

app.run(debug=True)

app.run()

**Index.html**

{% extends 'base.html' %}

{% block head %}

<link rel="stylesheet" href="{{url\_for('static',filename='css/main.css')}}">

{% endblock %}

{% block body %}

<script>

function myFunction1(){

window.location.href="/search";

}

</script>

<div class="topnav">

<h1>Restaraunt Recommendation</h1>

<a href="{{url\_for('home')}}">Home </a>

<a href="{{url\_for('search')}}">Recommend</a>

</div>

<br>

<br>

<p><img src="https://images.unsplash.com/photo-1517248135467-4c7edcad34c4?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=1170&q=80" alt="img">

<div class="bx">

This case study covers a very important business problem which is recommender systems. as we are in rapid consumption of content and commodities ordered by online delivery apps.

Challenges for these apps increase as they always want to keep their customer happy and wants to reduce their time. One major place where customer spends lots of their time is in searching for products.

Sometimes a consumer may know what product they are willing to buy but mostly don’t. Remember last time you ordered food online what took you most time whether it was to order your food, make the payment

or it was to hunt for the best vendor from where you can order your food or which caters to all your taste buds. mostly your answer will be you struggle most in hunting for the best vendors or restaurants.

Now imagine the last time when you wished to order some clothes and what you were aware of is that you wish to get a shirt/top with polka dots as they were in fashion but once you are on amazon you struggle

in finding the best design as there may be more than millions of polka dot pattern shirt in that case what you prefer is scrolling to recommended part and 40% of times you end up ordering items from recommended

tabs.

So this problem accounts for major purchases in e-commerce and e-content-based apps, not just these 2 the list goes on and on.

</p></div>

<input type="button" value="Test Now" onclick="myFunction1()"></input>

{% endblock %}