“RESTAURANT RECOMMENDATION SYSTEM”

Project report Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

in

CSE (DATA SCIENCE)

By

P. AJEETH KUMAR	**(20P71A6703)**
P. SAI RAGHAV	**(20P71A6734)**
V. LIKHITH	**(20P71A6748)**
M. ANVESH	**(21P75A6701)**

Under the Esteemed Guidance of

Mr. B. JOGA RAO, Assistant Professor

Department of CSE (DATA SCIENCE)

SWAMI VIVEKANANDA INSTITUTE OF TECHNOLOGY

Mahbub College Campus, Secunderabad-500003

(Affiliated to JNTU-H)

2020-2024

Approved by AICTE & Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the project report entitled **"RESTAURANT RECOMMENDATION SYSTEM"** is being submitted by M. ANVESH (21P75A6701), P. AJEETH KUMAR (20P71A6703), V. LIKHITH (20P71A6748), P. SAI RAGHAV (20P71A6734) in partial fulfillment for the award of Degree of BACHELOR OF TECHNOLOGY in CSE (DATA SCIENCE) to the Jawaharlal Nehru Technological University is a record of bonafide work carried out by him/her under my guidance and supervision.

Date:

Internal Guide

HOD-CSE (DATA SCIENCE)

External Examiner

Principal

ACKNOWLEDGEMENT

First of all, we thank our project guide Ms. B. Joga Rao Assistant Professor, and HOD Mrs. M. Supriya Samuel of the Department of CSE (Data Science) for giving us this opportunity in developing this project. This project has really helped us in enhancing our skills of programming, the perspective with which we should view projects and above all, our presentation and interpersonal skills. Last but not the least, we also thank the professors of our department for lending their helping hand whenever it was necessary.

P. AJEETH KUMAR
(20P71A6703)

P. SAI RAGHAV
(20P71A6734)

V. LIKHITH
(20P71A6748)

M. ANVESH
(21P75A6701)

DECLARATION

We hereby declare that the work which is being presented in this Mini Project entitled, **"RESTAURANT RECOMMENDATION SYSTEM"** submitted to **JNTU-H**, in the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **CSE (DATA SCIENCE)**, is an authentic record of my own work carried out from August 2023 to December 2023 under the supervision of **Mr. B. Joga Rao, Assistant Professor, CSE (DATA SCIENCE) Dept., SVIT, Mahbub Campus.**

The matter embodied in this project report has not been submitted by me for the award of any other degree.

Place: SECUNDERABAD
Date:

P. AJEETH KUMAR
(20P71A6703)

P. SAI RAGHAV
(20P71A6734)

V. LIKHITH
(20P71A6748)

M. ANVESH
(21P75A6701)

INDEX

ABSTRACT

In the dynamic landscape of real-world applications, this project introduces a cutting-edge machine learning-based restaurant recommendation system meticulously designed for practical impact. As the culinary world continues to evolve with diverse and discerning tastes, this innovative system leverages advanced algorithms to analyze user preferences, historical dining behaviors, and various contextual factors. Tailoring its recommendations to individual preferences, the system goes beyond a generic approach, ensuring a personalized and enhanced dining experience for users. At the core of this sophisticated application is the utilization of machine learning techniques, allowing the system to unravel intricate patterns within vast datasets. By delving into the nuances of individual dining behaviors, the recommendation system gains a comprehensive understanding of user preferences, spanning not only culinary tastes but also factors such as ambiance, dietary restrictions, and pricing considerations. This analytical prowess enables the system to adapt continually, ensuring that recommendations remain aligned with the dynamic nature of culinary trends and individual preferences. Furthermore, the recommendation system integrates location-based services, optimizing its suggestions by considering the geographical proximity of users to various dining establishments. This strategic inclusion enhances the accessibility and convenience for users, making the dining experience not only personalized but also tailored to their immediate surroundings. Real-time data, including user reviews and ratings, adds another layer of sophistication, allowing the system to discern the latest trends and the popularity of specific venues. In the broader context of the real-world application, this recommendation system serves as a practical tool for simplifying decision-making for users and contributes significantly to the vibrancy of the restaurant industry. By fostering informed choices and encouraging diverse culinary exploration, the system emerges as a key enabler in shaping consumer behaviors and preferences. As we delve into the intricacies of this innovative project, it becomes evident that this restaurant recommendation system stands at the forefront of technological advancements, seamlessly connecting consumers with dining establishments and enriching the gastronomic journey for all.

LIST OF FIGURES

1. INTRODUCTION

The advancement of science and technology has significantly improved the quality of life, particularly in the context of emerging technologies related to transportation, uncertainty resolution, fuzzy shortest paths, PowerShell, wireless sensor networks, computer languages, neural networks, routing, and image processing. These technologies contribute to the development of intelligent and self-healing products. The integration of smart city applications such as smart water management, smart grids, smart parking, and smart resource management is heavily reliant on Internet of Things (IoT) and Internet of Everything (IoE) technologies. This manuscript focuses on the development of a recommendation system for restaurants, leveraging criteria such as cast, keywords, crew, and genres. The primary objective of the recommendation system is to predict users' interests and suggest items that are likely to be of interest to them. The escalating volume of information available online, coupled with the growing number of Internet users, has led to information overload, making it challenging to find relevant information promptly. The recommender system addresses this challenge by filtering data from a vast pool based on user preferences or interests. These systems play a pivotal role in suggesting restaurants, creating personalized dining experiences, and catering to user preferences. Recommender systems in the restaurant context operate based on characteristic information about users and items, as well as user-item interactions. Characteristic information includes details about the user and the restaurants, while user-item interactions encompass ratings, number of visits, likes, and other relevant user activities. The recommendation system can be developed using various techniques such as collaborative filtering, content-based filtering, or a hybrid approach, ensuring a tailored and enjoyable dining experience for users.

2. SYSTEM REQUIREMENTS AND ANALYSIS

2.1 Problem Definition

Creating a recommender system for restaurants from the ground up presents unique challenges. In cases where a website lacks a substantial user base, a common scenario in early stages, conventional user-based recommender systems may be less effective. In such instances, alternative strategies need consideration to ensure accurate recommendations despite limited user data.

2.1.1 Disadvantages of Existing Systems:

The existing system proposes lot of restaurants without any knowledge and without users interest.

2.2 Problem Analysis

While conventional content-based recommendation systems typically cater to user preferences within established parameters, our project takes a novel approach by employing a content-based system to offer unique and unconventional suggestions. This system utilizes attributes such as cuisine type, chef, restaurant description, notable features, etc., to generate recommendations for users. The underlying idea is that if a user enjoyed a particular restaurant, they might appreciate recommendations for similar dining experiences. This adaptation of content-based methodology extends beyond conventional boundaries, providing users with diverse and out-of-the-box restaurant suggestions based on shared attributes and preferences.

2.2.1 Advantages of proposed System:

- It saves time and human effort.

- Using our recommender system makes user to find similar items.

2.3 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system are essential.
Three key considerations involved in the feasibility analysis are
1. Economical Feasibility
2. Technical Feasibility
3. Social Feasibility

Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.4 System Requirements

Hardware Requirements

Processor	:	Intel Core i5
Hard Disk	:	250GB
RAM	:	16GB

Software Requirements

Operating systems	:	Windows 10
Technologies	:	Python, Jupyter Notebook
Libraries	:	Pandas, Numpy, Matplotlib, NLTK, Sci-kit Learn

3. SYSTEM DESIGN

3.1 INTRODUCTION TO UML

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows:

User Model View

In this module admin maintains the database and provides the details of the movies.

Structural Model View

In this model, the data and functionality are arrived from inside the system. This model view models the static structures.

Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View

In this view, the structural and behavioral as parts of the system are represented as they are to be built.

Environmental Model View

In this view, the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

3.2 UML Diagrams

3.2.1 Class diagram

Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship in between them. The main purpose to use class diagrams are:

1. This is the only UML which can appropriately depict various aspects of OOP's concept.
2. Proper design and analysis of application can be faster and efficient.
3. It is base for deployment and component diagram.
4. Each class is represented by a rectangle having a subdivision of three: Compartments name, attributes and operation.

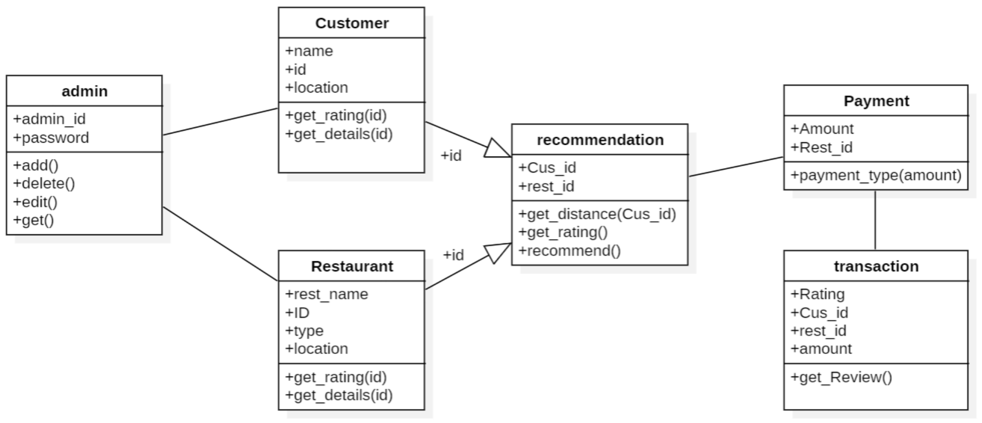


Figure 3.2.1. Class Diagram

The flowchart illustrates a streamlined restaurant ordering process for a mobile app, accommodating both registered and unregistered users. Registered users initiate by logging in, selecting a nearby restaurant, choosing dishes, online payment, and real-time order tracking. Unregistered users follow a similar path, with an added step for entering contact information. Error conditions, such as dish unavailability or payment issues, are seamlessly addressed. This meticulous flowchart ensures a user-friendly and error-tolerant ordering experience.

3.2.2 Use-Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. To clarify a bit in details, dynamic behavior means the behavior of the system when it is running/operating.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

a. Used to gather requirements of a system.
b. Used to get an outside view of a system.
c. Identify external and internal factors influencing the system.
d. Show the interacting among the requirements are actors

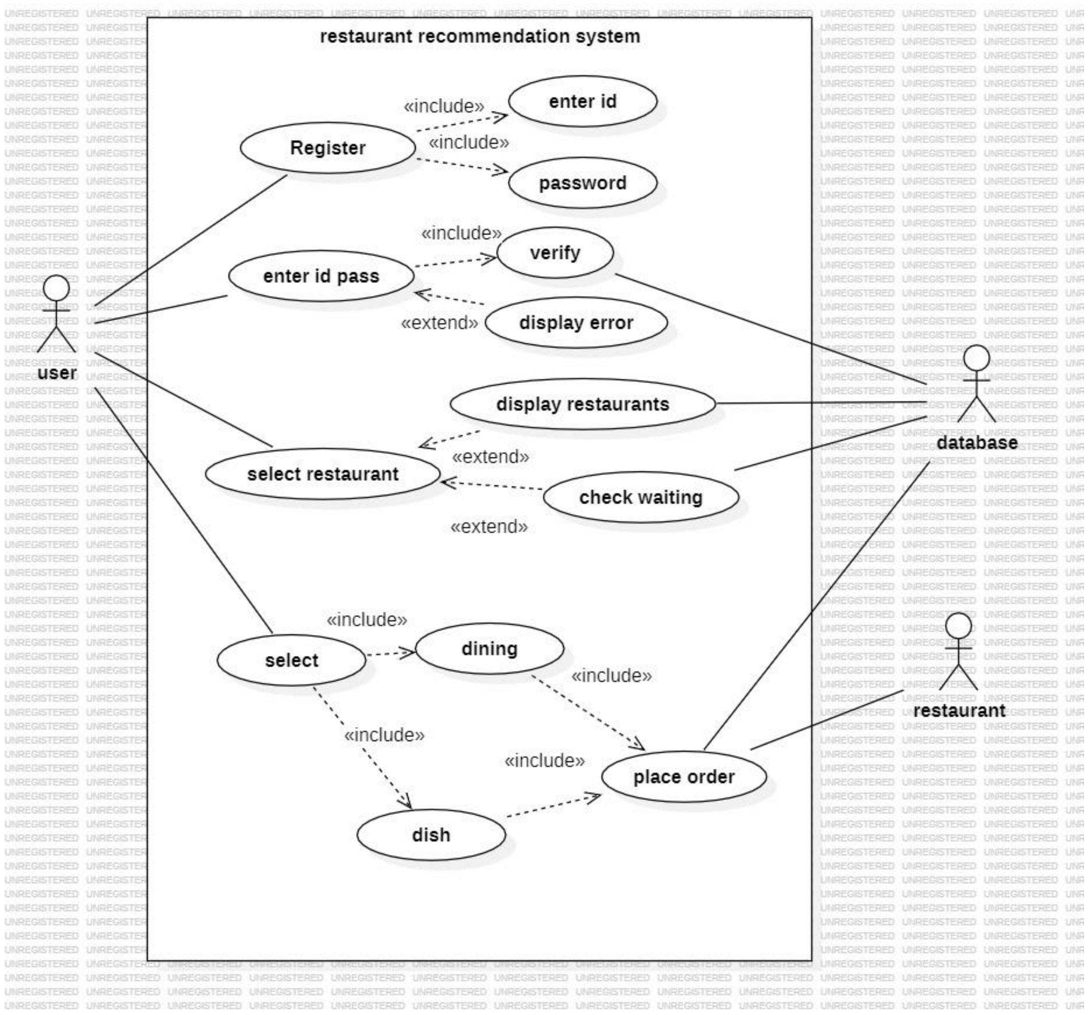


Figure 3.2.2 Use-Case Diagram

The presented flowchart delineates the sequential steps that customers can undertake to place a food order at a restaurant, encompassing the process from restaurant selection to order placement. The flowchart is bifurcated into two principal segments: one tailored for registered users and another for unregistered users. Registered users are afforded the option to log in to their accounts, facilitating access to their order history, while unregistered users are provided with the option to order as guests. The initial step for both registered and unregistered users is the selection of a restaurant. Subsequently, users can peruse the menu, make selections of desired items, and progress to the checkout stage.
During checkout, users are required to input their payment details and delivery information. Following the placement of the order, it is transmitted to the kitchen, and users receive notifications when their food is ready.

3.2.3 Sequence diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

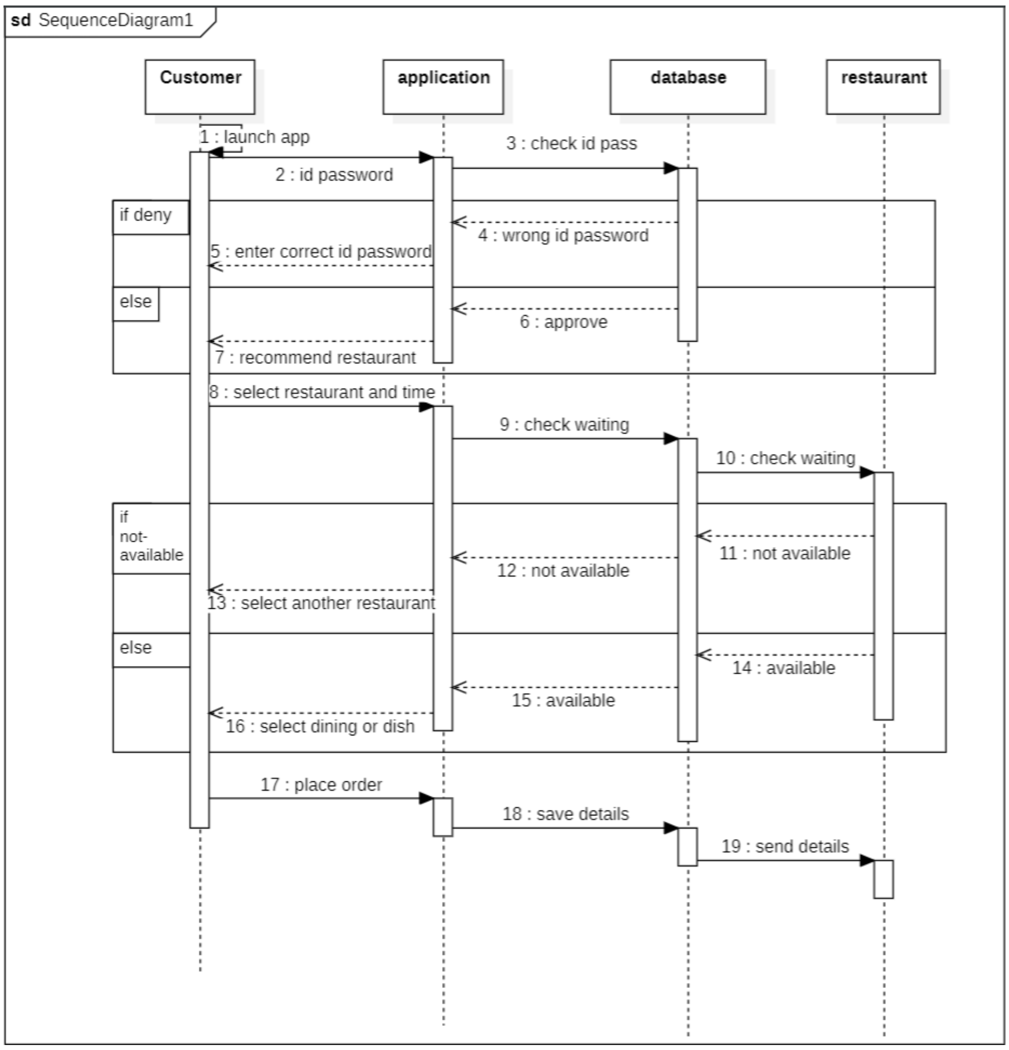


Figure 3.2.3 Sequence Diagram

The flowchart depicts the user journey in a restaurant reservation app. For registered users, the process starts with login or registration, followed by restaurant selection, availability check, table booking, and confirmation. Unregistered users follow a similar path but register before confirmation. Additional features include menu viewing, special requests, and reservation management options such as cancellations and feedback provision. The flowchart ensures a concise overview of the app's functionality for both registered and unregistered users.

3.2.4 Activity diagram

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows

1. Identify candidate use cases, through the examination of business workflows

2. Identify pre- and post-conditions (the context) for use cases

3. Model workflows between/within use cases

4. Model complex workflows in operations on objects

5. Model in detail complex activities in a high level activity Diagram

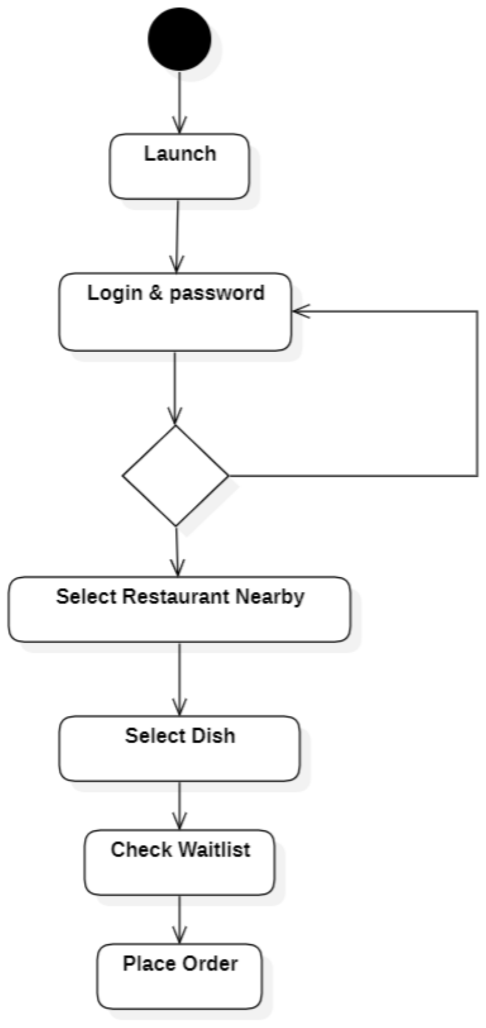


Figure 3.2.4 Activity Diagram

The flowchart outlines the restaurant ordering process for a mobile app, catering to both registered and unregistered users. Registered users start with login, restaurant selection, and menu browsing, followed by order confirmation, online payment, and real-time tracking. Unregistered users, after entering the app, choose delivery or eat-in, proceed with restaurant and menu selection, provide contact information for order confirmation, and complete the process with online payment. The flowchart includes error handling for contingencies like dish unavailability or payment issues. Additional features comprise order history viewing, account settings management, and optional order tracking for delivery.

4. IMPLEMENTATION DETAILS

4.1 Functional Requirements

In Software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation, processing, and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements. Generally, functional requirements are expressed in the form "**system shall do <requirement>".** In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirement analyst generates use cases after gathering and validating a set of functional requirements.

4.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours

The non-functional requirements are

- **Availability**

A system's availability is the amount of time that is operational and available for use.

- **Efficiency**

Specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth etc.

- **Flexibility**

If the organization intends to increase or extend the functionality of the software after it is deployed, that should be planned from the beginning; it influences choices made during the design, development, testing and deployment of the system. New modules can be easily integrated to our system without disturbing the existing modules or modifying the logical database schema of the existing applications.

- **Portability**

Portability specifies the case with which the software can be installed on all necessary platforms, and the platforms on which it is expected to tun. This allows the application to be easily operated on any operating system.

- **Scalability**

Software that is scalable has the ability to handle a wide variety of system configuration sizes. The non-functional requirements should specify the ways in which the system may be expected to scale up (by increasing hardware capacity, adding machines etc.). Our system can be easily expandable. Any additional requirements such as hardware or software which increase the performance of the system can be easily added.

- **Integrity**

Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software.

- **Usability**

Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and widely used by its intended users.

- **Performance**

The performance constraints specify the timing characteristics of the software.

4.3 Software Used:

➢ PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

➢ DJANGO

Django is a high-level Python web framework that facilitates the rapid development of web applications, including those for data science and machine learning. It seamlessly integrates with popular Python libraries such as scikit-learn, Keras, PyTorch, SymPy (LaTeX), NumPy, pandas, and Matplotlib.

➢ PANDAS

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data managing and preparation.

➢ NUMPY

NumPy is an open-source Python library that excels in high-performance numerical computing and array manipulation. It introduces efficient data structures, particularly the powerful NumPy array, making it a fundamental tool for tasks like linear algebra and statistical analysis. With seamless integration into Python's data ecosystem, NumPy enhances the language's capabilities in numerical computing, forming a critical component alongside libraries like Pandas for effective data manipulation and analysis.

➢ MATPLOTLIB

Matplotlib is an open-source Python library which is a versatile tool for creating static, animated, and interactive visualizations. Widely used in data science and research, Matplotlib offers a comprehensive set of plotting functions, making it easy to generate diverse charts and graphs. Known for its simplicity and compatibility with different platforms, Matplotlib is a go-to choice for creating clear and visually appealing visualizations, ranging from basic line plots to intricate 3D graphics.

➢ NLTK

NLTK (Natural Language Toolkit) is a powerful open-source library for working with human language data in Python. Designed for tasks such as text processing, tokenization, part-of-speech tagging, and sentiment analysis, NLTK provides a comprehensive suite of tools for natural language processing (NLP). With a vast collection of corpora, lexical resources, and pre-trained models, NLTK facilitates the development of applications related to language understanding and analysis. Its user-friendly interface makes it accessible for both beginners and experienced developers, positioning NLTK as a leading library in the field of natural language processing within the Python ecosystem.

➢ SCIKIT-LEARN

Scikit-learn is a user-friendly and efficient open-source machine learning library for Python. With a straightforward API, it offers tools for classification, regression, clustering, and more. Known for its versatility, it seamlessly integrates with other Python libraries, making it a foundational choice for various machine learning applications.

Pseudo code

```
Step 1: Import the required packages

Step 2: Parse command-line arguments.

Step 3: Load model.

Step 4: Upload data

Step 5: Get data
```

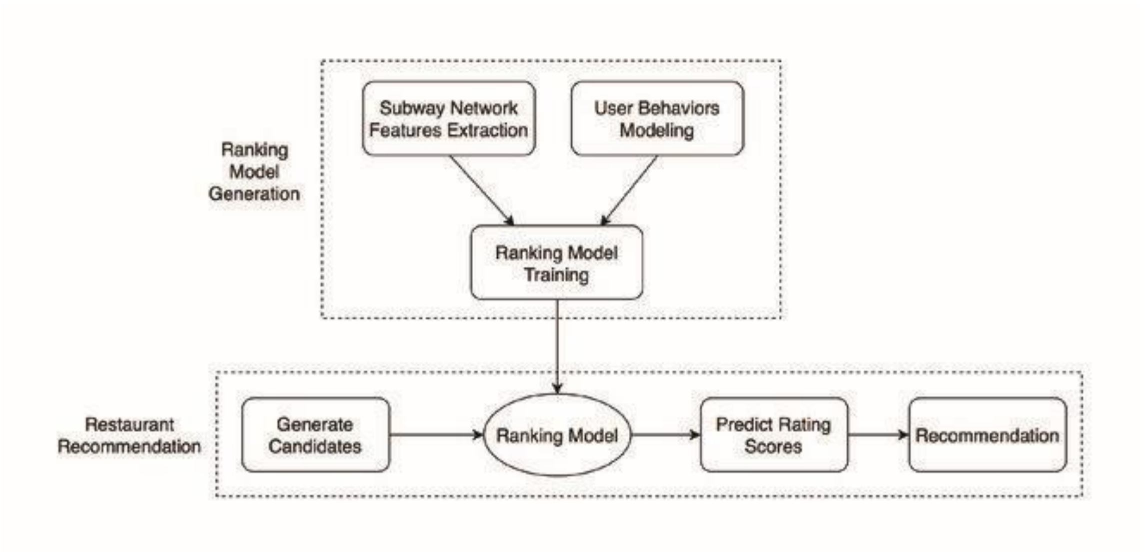


Fig 4.1: Workflow Diagram

5. CODE SNIPPETS

JUPYTER NOTEBOOK CODE

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import re
sns.set_style('darkgrid')

#reading the dataset
df=pd.read_csv("zomato.csv")
df.head() # prints the first N rows of a DataFrame
#Deleting Unnnecessary Columns
df=df.drop(['url','dish_liked','phone','address','menu_item','location'],axis=1)
df.duplicated().sum()
df.drop_duplicates(inplace=True)
#Remove the NaN values from the dataset
df.isnull().sum()
df.dropna(how='any',inplace=True)
df.info() #.info() function is used to get a concise summary of the dataframe
df=df.rename(columns={"name":'Name','rate':'Ratings','votes':'Votes','rest_type' :
'Restaurant_Type','cuisines':'Cuisines', 'approx_cost(for two people)' : 'Cost',
'listed_in(type)' : 'Type','listed_in(city)' : 'City', 'online_order' : 'Online
Orders','book_table':'Booking Table','reviews_list' : 'Review'})
df['Cost'].unique()
#Some Transformations
df['Cost'] = df['Cost'].astype(str) #Changing the Cost to string
df['Cost'] = df['Cost'].apply(lambda x: x.replace(',','')) #Using lambda function to
replace ','
```

```
    from Cost
    df['Cost'] = df['Cost'].astype(float) # Changing the Cost to Float
    df.info()
    ## function to remove commas and convert the values
    ## into numbers
    def Cost(value):
        value = str(value)
        if "," in value:
            value = float(value.replace(",",""))
  return value
else:
  return float(value)
    df['Cost'] = df['Cost'].apply(Cost)
    print(df['Cost'].head())
    #Removing '/5' from Rates
    df = df.loc[df.Ratings !='NEW']
    df = df.loc[df.Ratings !='-'].reset_index(drop=True)
    remove_slash = lambda x: x.replace('/5', '') if type(x) == str else x
    df.Ratings = df.Ratings.apply(remove_slash).str.strip().astype('float')
    df['Ratings'].head()
    # Adjust the column names
    df.name = df.Name.apply(lambda x:x.title())
    df['Online Orders'].replace(('Yes','No'),(1, 0),inplace=True)
    df['Booking Table'].replace(('Yes','No'),(1, 0),inplace=True)
    df.Cost.unique()
    restaurants = list(df['Name'].unique())
    df['Mean Rating'] = 0

    for i in range(len(restaurants)):
        df['Mean Rating'][df['Name'] == restaurants[i]] = df['Ratings'][df['Name'] ==
        restaurants[i]].mean()
    ## Lower Casing
```

```
      mini_project["Review"] = mini_project["Review"].str.lower()
      mini_project[['Review', 'Cuisines']].sample(5)
      def extract_ratings(text):
            # Use regular expression to find numbers with a decimal point
  ratings = re.findall(r'(\d+\.\d+)', text)
  if ratings:
      return float(ratings[0])  # Convert the extracted rating to a float
   else:
      return None  # Return None if no rating is found

      # Apply the function to the 'text' column and store results in a new column 'ratings'
      mini_project['User_Ratings'] = mini_project['Review'].apply(extract_ratings)
      print(mini_project.isnull().sum())
      #Remove the NaN values from the dataset
      mini_project.isnull().sum()
      mini_project.dropna(how='any',inplace=True)
      mini_project.info() #.info() function is used to get a concise summary of the
dataframe
      mini_project = mini_project.drop(['Review'],axis=1)
      print(mini_project['Online Orders'].value_counts())
      plt.figure(figsize=(30,10))
      mini_project['Online Orders'].value_counts().plot(kind='pie',
      colors=['lightblue','skyblue'],autopct='%1.1f%%', textprops={'fontsize': 15})
      plt.title('% of restaurants that take online orders',size=20)
      plt.xlabel('',size=15)
      plt.ylabel('',size=15)
      plt.legend(loc=2, prop={'size': 15})
      print(mini_project['Booking Table'].value_counts())
      plt.figure(figsize=(30,10))
      mini_project['Booking Table'].value_counts().plot(kind='pie',
      colors=['plum','mediumorchid'], autopct='%1.1f%%', textprops={'fontsize': 15})
      plt.title('% of restaurants that provide table booking facility',size=20)
      plt.xlabel('',size=15)
      plt.ylabel('',size=15)
      plt.legend(loc=2, prop={'size': 15})
      ratings=mini_project.groupby(['Ratings']).size().reset_index().rename(columns={0:
"Rating_Count"})
      plt.figure(figsize=(30,10))
      sns.barplot(x='Ratings',y='Rating_Count',data=ratings)
      plt.title('Rating vs Rating counts',size=30)
      plt.xlabel('Ratings',size=30)
      plt.ylabel('Ratings Count',size=30)
      plt.figure(figsize=(30,10))
      sns.lmplot(x='Ratings',y='Cost',data=mini_project,height=7)
      plt.xlabel('Ratings',size=15)
      plt.ylabel('Cost for two people',size=15)
      plt.xticks(fontsize=15)
      plt.yticks(fontsize=15)
      current_values = plt.gca().get_yticks()
      plt.gca().set_yticklabels(['{:,.0f}'.format(x) for x in current_values])
```

DJANGO CODE

```
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
```

```
    SECRET_KEY='django-insecure
ul)jvuib!!3#@2p*kf5#l8k0s1#kay!rpi975x$k7rges0c7#v'
    # SECURITY WARNING: don't run with debug turned on in production!
    DEBUG = True

    ALLOWED_HOSTS = []

    # Application definition

    INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'restaurant',
    ]

    MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    ]

    ROOT_URLCONF = 'rr.urls'

    TEMPLATES = [
 {
     'BACKEND': 'django.template.backends.django.DjangoTemplates',
     'DIRS': [os.path.join(BASE_DIR,'templates')],
     'APP_DIRS': True,
     'OPTIONS': {
         'context_processors': [
             'django.template.context_processors.debug',
             'django.template.context_processors.request',
             'django.contrib.auth.context_processors.auth',
             'django.contrib.messages.context_processors.messages',
             ],
         },
     },
 ]

WSGI_APPLICATION = 'rr.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
{
   'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
   'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
   'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
   'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]


# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/
```

```
STATIC_URL = 'static/'

#added manually
STATICFILES_DIRS = [
    os.path.join(BASE_DIR,'static'),
]
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

6. METHODOLOGY

The ‘Zomato Restaurants Dataset’ is taken into consideration for restaurant recommendation purpose in this research work. This dataset is available on kaggle.com.

- The dataset is composed of 1 CSV file - ‘zomato.csv’

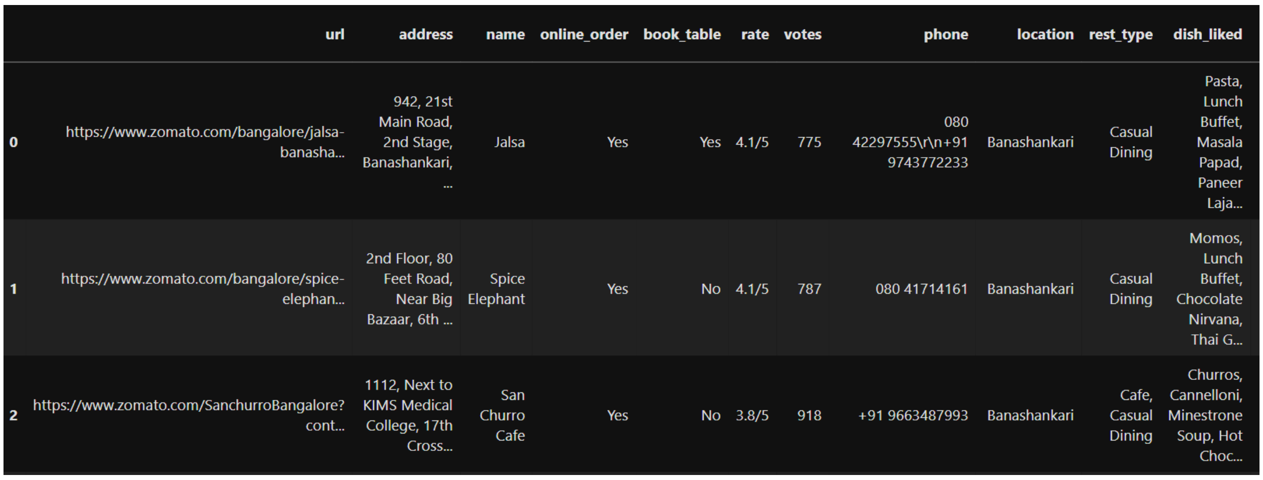
The ‘zomato.csv’ dataset consists of the following attributes:

- ‘url’: It indicates the url of the restaurant in the zomato website.
- ‘address’: It indicates the address of the restaurant in Bengaluru.
- ‘name’: It indicates the name of the restaurant.
- ‘online_order’: It indicates online ordering is available in the restaurant or not.
- ‘book_table’: It indicates table book option available or not.
- ‘rate’: It indicates the overall rating of the restaurant out of 5.
- ‘votes’: It is the total number of rating for the restaurants mentioned.
- ‘phone’: It is the phone number of the restaurant.
- ‘location’: It is the neighborhood in which the restaurant is located.
- ‘rest_type’: It consists of the restaurant type.

Perform Exploratory Data Analysis (EDA) on the data:

The dataset contains two CSV files, credits, and movies. The credits file contains all the metadata information about the movie and the movie file contains the information like name and id of the movie, budget, languages in the movie that has been released, etc.

```
import numpy as np
import pandas as pd
#reading the dataset
df=pd.read_csv("zomato.csv")
df.head() # prints the first N rows of a DataFrame
```



	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked
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...
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...
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	Churros, Cannelloni, Minestrone Soup, Hot Choc...

Fig 6.1: Restaurants List

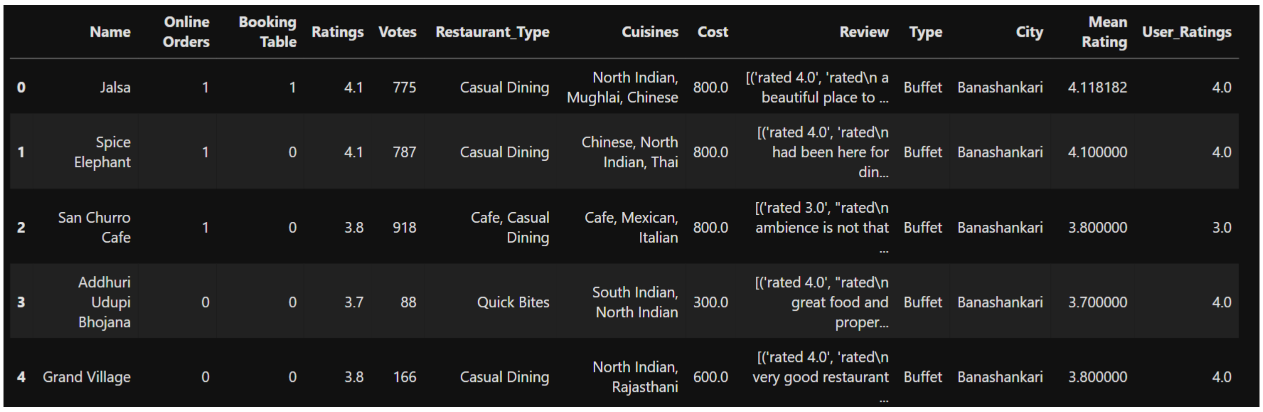
The presented visual excerpt encapsulates a tabular representation featuring pertinent details of various restaurants. Noteworthy elements encompass essential information such as restaurant names, addresses, contact numbers, and additional particulars. Specific restaurants identified within this dataset include Jalsa, Spice Elephant, San Churro, Addhuri Bhojana, and Grand Village.

The table is indicative of the platform's diverse functionalities, as denoted by columns titled "online order" and "book table," suggesting a dual capacity for facilitating both online food orders and restaurant reservations. Furthermore, supplemental details are provided through columns such as "rate," indicative of the average customer rating, "votes," representing the quantity of ratings received, "rest_type" denoting the classification of the restaurant, "dish_liked," potentially signifying a popular dish, and "location," likely specifying the geographic area in which the restaurant is situated.

Upon comprehensive examination, it can be inferred that the image portrays a comprehensive listing of restaurants accessible on a food delivery or reservation platform, specifically within the locale of Bangalore, India. Users navigating this platform can seamlessly peruse restaurant options, evaluate delivery and dining alternatives, and glean additional insights such as customer ratings and favored dishes.

```
df.info()
df['listed_in(city)'].unique()
#Deleting Unnnecessary Columns
df=df.drop(['url','dish_liked','phone','address','menu_item','location'],axis=1)
df.duplicated().sum()
df.drop_duplicates(inplace=True)
#Remove the NaN values from the dataset
df.isnull().sum()
df.dropna(how='any',inplace=True)
df.info() #.info() function is used to get a concise summary of the dataframe
print(mini_project.isnull().sum())

print(mini_project['Online Orders'].value_counts())
plt.figure(figsize=(30,10))
mini_project['Online
Orders'].value_counts().plot(kind='pie',colors=['lightblue','skyblue'],autopct='%1.1f%%',
textprops={'fontsize': 15})
plt.title('% of restaurants that take online orders',size=20)
plt.xlabel('',size=15)
plt.ylabel('',size=15)
plt.legend(loc=2, prop={'size': 15})
```



	Name	Online Orders	Booking Table	Ratings	Votes	Restaurant_Type	Cuisines	Cost	Review	Type	City	Mean Rating	User_Ratings
0	Jalsa	1	1	4.1	775	Casual Dining	North Indian, Mughlai, Chinese	800.0	[('rated 4.0', 'rated\n a beautiful place to ...	Buffet	Banashankari	4.118182	4.0
1	Spice Elephant	1	0	4.1	787	Casual Dining	Chinese, North Indian, Thai	800.0	[('rated 4.0', 'rated\n had been here for din...	Buffet	Banashankari	4.100000	4.0
2	San Churro Cafe	1	0	3.8	918	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	[('rated 3.0', "rated\n ambience is not that ...	Buffet	Banashankari	3.800000	3.0
3	Addhuri Udupi Bhojana	0	0	3.7	88	Quick Bites	South Indian, North Indian	300.0	[('rated 4.0', "rated\n great food and proper...	Buffet	Banashankari	3.700000	4.0
4	Grand Village	0	0	3.8	166	Casual Dining	North Indian, Rajasthani	600.0	[('rated 4.0', 'rated\n very good restaurant ...	Buffet	Banashankari	3.800000	4.0

Fig 6.2: Cleaned Restaurant List

```
print(mini_project['Online Orders'].value_counts())
plt.figure(figsize=(20,8))
mini_project['Online
Orders'].value_counts().plot(kind='pie',colors=['lightblue','skyblue'],autopct='%1.1f%%',
textprops={'fontsize': 15})
plt.title('% of restaurants that take online orders',size=20)
plt.xlabel('',size=15)
plt.ylabel('',size=15)
plt.legend(loc=2, prop={'size': 10})
```

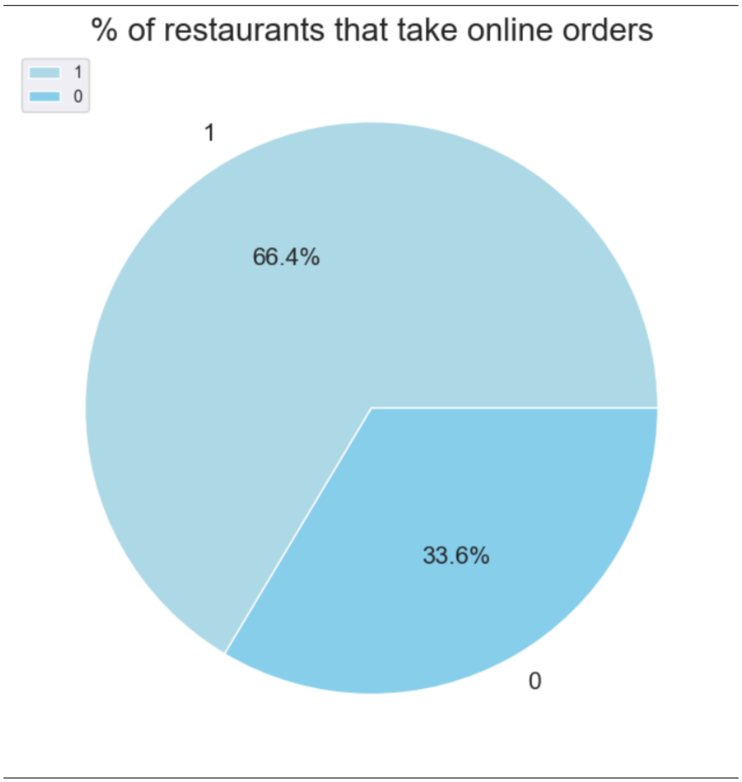


Fig 6.3: Online Orders

This pie chart indicates that a significant majority of restaurants, 66.4%, take online orders. This suggests that online ordering is a popular and convenient option for both customers and restaurants. There are a few possible reasons why the remaining 33.6% of restaurants do not take online orders. Some restaurants may be too small or lack the resources to set up online ordering systems. Others may prefer to focus on in-person dining experiences. Still others may be located in areas where online ordering is not as popular.

Overall, the pie chart suggests that online ordering is a growing trend in the restaurant industry. However, there is still a significant number of restaurants that do not offer this service.

```
print(mini_project['Booking Table'].value_counts())
plt.figure(figsize=(20,10))
mini_project['Booking
Table'].value_counts().plot(kind='pie',colors=['plum','mediumorchid'],autopct='%1.1f%
%', textprops={'fontsize': 15})
plt.title('% of restaurants that provide table booking facility',size=20)
plt.xlabel('',size=15)
plt.ylabel('',size=15)
plt.legend(loc=2, prop={'size': 15})
```

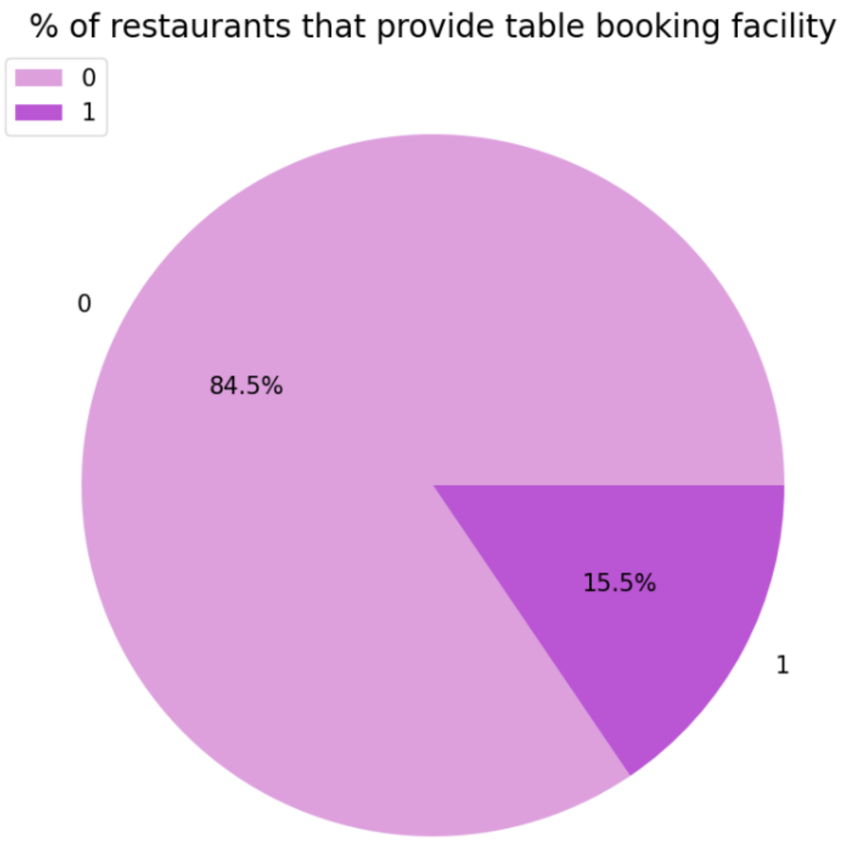


Fig 6.4: Table Facility

In overview, the pie chart illustrates a prevailing trend, revealing that a substantial 84.5% of the represented restaurants extend the provision of a table booking facility. This prominent indication suggests that a noteworthy majority within the dataset afford patrons the convenience of reserving tables in advance.

The delineation of the pie chart comprises two distinct segments:

The larger, green-colored segment, constituting 84.5%, signifies the proportion of restaurants that do provide a table booking option.
The smaller, orange-colored segment, representing 15.5%, denotes the percentage of restaurants that do not offer a table booking facility.
Augmenting clarity, the pie chart incorporates text labels, including an overarching title: "% of Restaurants That Provide Table Booking Facility." Additionally, each respective segment is annotated with its corresponding percentage value, distinctly presented as 84.5% and 15.5%.

7. TESTING

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 Types of testing

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

The following are the types of Integration Testing:

1. Top-down Integrating Testing: This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through

the control hierarchy, beginning with the main program module. The module subordinates to the main program module which is incorporated into the structure in wither depth first or breadth first manner.

2. Bottum-up Integration Testing: This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to give level is always available and the needs for stubs is eliminated.

Functional Testing: Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is dependent on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

7.2 Black Box Testing

What is Black Box Testing?

Black box testing is a software testing technique in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focuses on inputs and outputs of the software system without bothering about internal knowledge of the software program.

Input
Black Box
Output

Fig 7.1 Black box testing

The above Black Box can be any software system you want to test. For example; an operating system like Windows, a website like Google, a database like Oracle or even our own custom application. Under Black Box testing, we can test these applications by just focusing on the inputs and outputs without knowing their internal code implementations.

Black box testing – Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes
  them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones

- **Functional testing** – This black box testing type is related to functional requirements of a system, it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression testing is done after codes fixes, upgrades pr maintenance to check whether the new code has not affected the existing code.

7.3 White Box Testing

White Box testing of software solution’s internal ending and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as clear,open, structural, and glass box testing.

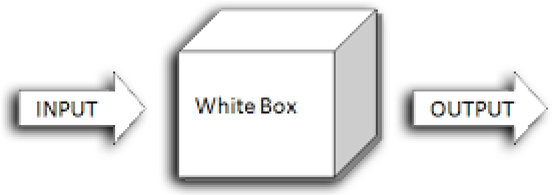


Figure 7.2 White Box Testing

It is one of parts of “box testing” approach of software testing. Its counterpart black box testing from an external or end-user type perspective. On the other hand, White box testingis based on the inner workings of an application and revolves around internal testing. Theterm “white box” was used because of the see-through box concept. The clear box or white box names symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “black box testing” symbolizes not being able to see the inner workings of the software so that the only end-user experience can be tested.

- **What do we verify in White Box Testing?**

White box testing involves the testing of the software code for the following:Internal security holes.

- ➢ Broken or poorly structured paths in the coding processes.
- ➢ The flow of specific inputs through the code.
- ➢ Expected output.
- ➢ The functionality of conditional loops.
- ➢ Testing of each statement object and function on an individual basis.
- ➢ Control flow testing examines the flow of control between different statements, loops, and conditional statements.

The testing can be done at system, integration and unit levels of software development. One of the basic goals of white box testing is to verify a working flow for an application.It involves testing a series of predefined inputs against expected or desired outputs so thatwhen a specific input does not result in the expected output, you have encountered a bug.

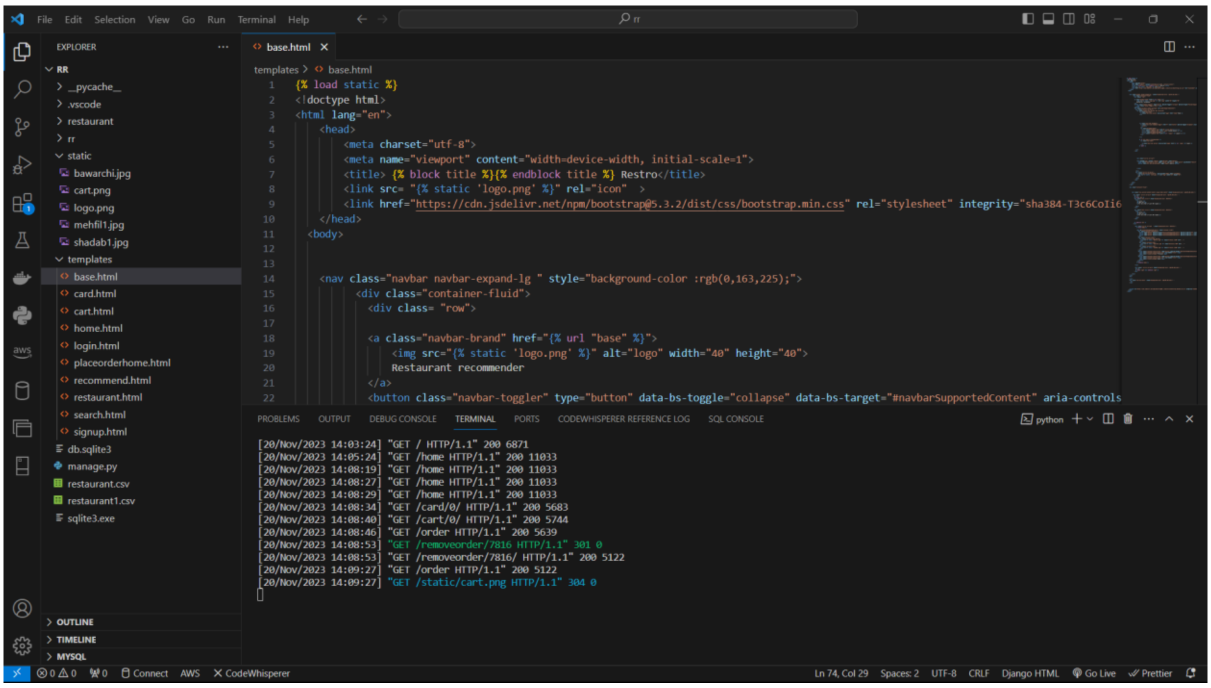
7.4 Test Cases:

Test cases are an integral part of software testing, playing a crucial role in ensuring the quality and reliability of a software application. They are sets of conditions or variables under which a tester will determine whether a system or specific features of an application are working as intended. The primary objective of test cases is to validate that the software meets the specified requirements and functions correctly in various scenarios. Well-designed and comprehensive test cases are essential for ensuring the quality and reliability of software. They contribute to the overall success of the development process by identifying and resolving issues early in the lifecycle, ultimately delivering a more robust and user-friendly product.

Test Cases

Test Case ID	Test Case	Expected Output	Actual Output	Result
T1	Predict Restaurant Using Model	Predicted Restaurant	The restaurant is predicted	Pass
T2	Predict Restaurant Using Search	Predicted Result	The restaurant searched is printed.	Pass

8. OUTPUT SCREENS



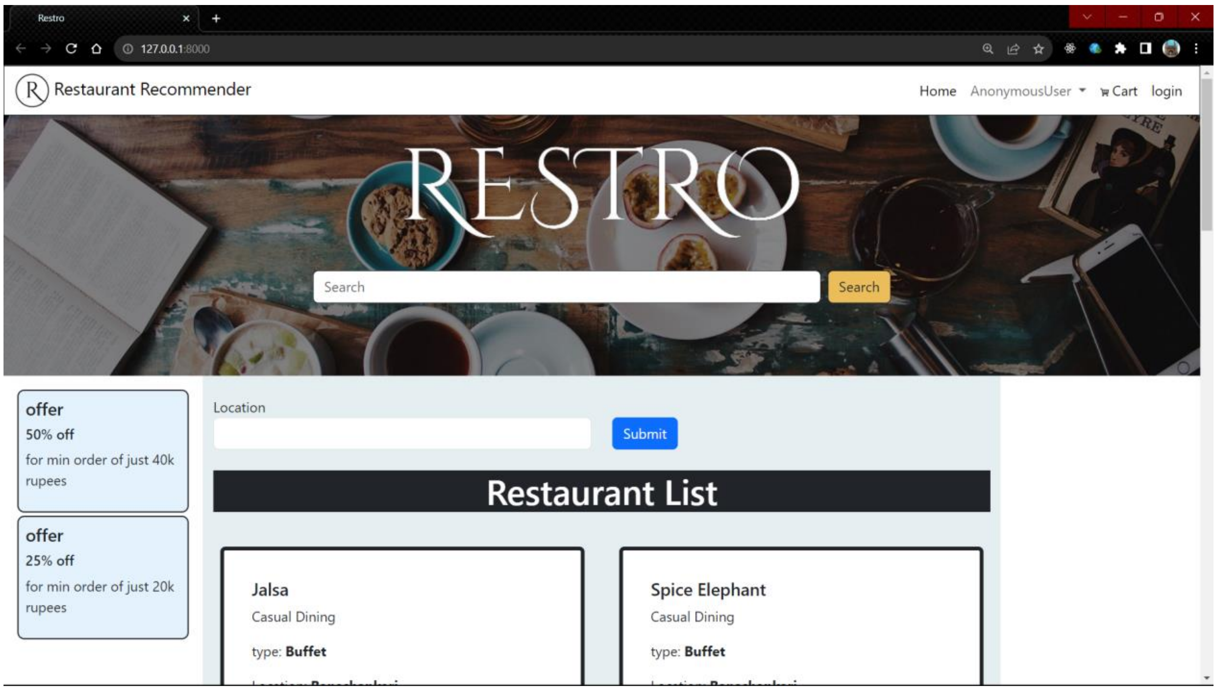
```
{% load static %}
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title> {% block title %}{% endblock title %} Restro</title>
        <link src= "{% static 'logo.png' %}" rel="icon"  >
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6
    </head>
  <body>


    <nav class="navbar navbar-expand-lg " style="background-color :rgb(0,163,225);">
          <div class="container-fluid">
            <div class= "row">

            <a class="navbar-brand" href="{% url "base" %}">
                <img src="{% static 'logo.png' %}" alt="logo" width="40" height="40">
                Restaurant recommender
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls
```

```
[20/Nov/2023 14:03:24] "GET / HTTP/1.1" 200 6871
[20/Nov/2023 14:05:24] "GET /home HTTP/1.1" 200 11033
[20/Nov/2023 14:08:19] "GET /home HTTP/1.1" 200 11033
[20/Nov/2023 14:08:27] "GET /home HTTP/1.1" 200 11033
[20/Nov/2023 14:08:29] "GET /home HTTP/1.1" 200 11033
[20/Nov/2023 14:08:34] "GET /card/0/ HTTP/1.1" 200 5683
[20/Nov/2023 14:08:40] "GET /cart/0/ HTTP/1.1" 200 5744
[20/Nov/2023 14:08:46] "GET /order HTTP/1.1" 200 5639
[20/Nov/2023 14:08:53] "GET /removeorder/7816 HTTP/1.1" 301 0
[20/Nov/2023 14:08:53] "GET /removeorder/7816/ HTTP/1.1" 200 5122
[20/Nov/2023 14:09:27] "GET /order HTTP/1.1" 200 5122
[20/Nov/2023 14:09:27] "GET /static/cart.png HTTP/1.1" 304 0
```

Figure 8.1 Execution



Restaurant Recommender

Home AnonymousUser Cart login

RESTRO

Search Search

offer
50% off
for min order of just 40k rupees

offer
25% off
for min order of just 20k rupees

Location

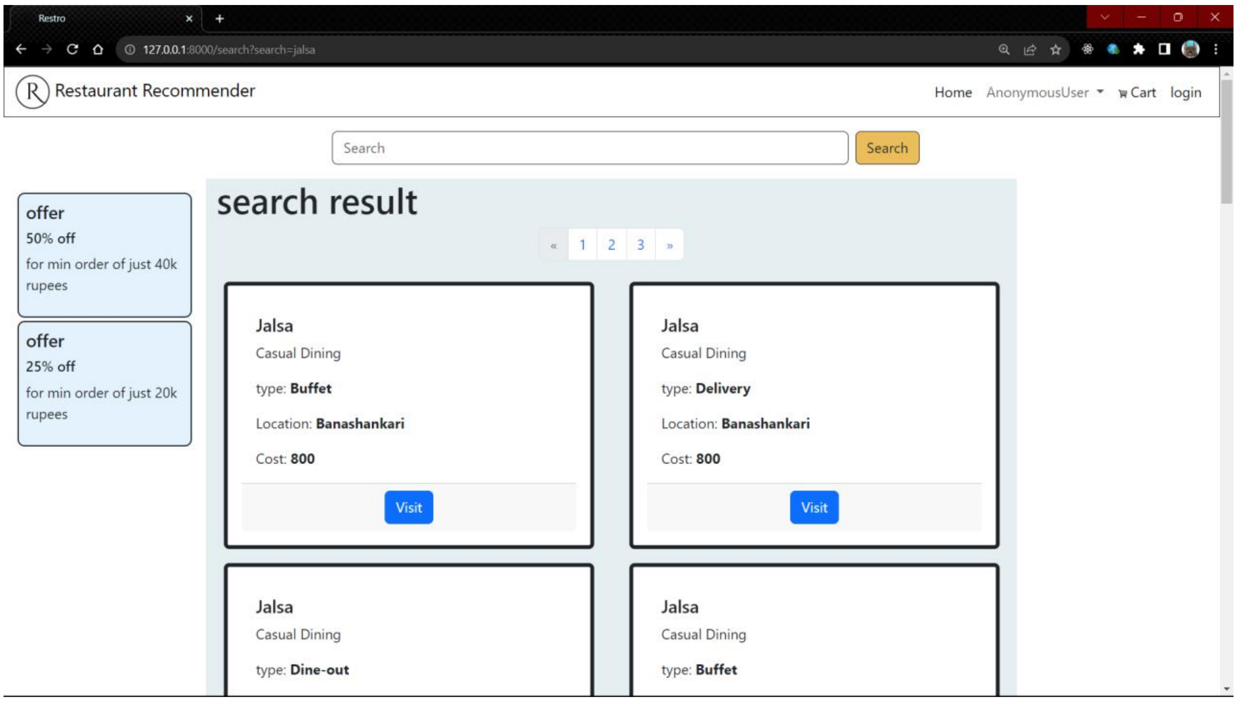
Submit

Restaurant List

Jalsa
Casual Dining
type: **Buffet**

Spice Elephant
Casual Dining
type: **Buffet**

Figure 8.2 UI screen



search result

offer
50% off
for min order of just 40k rupees

offer
25% off
for min order of just 20k rupees

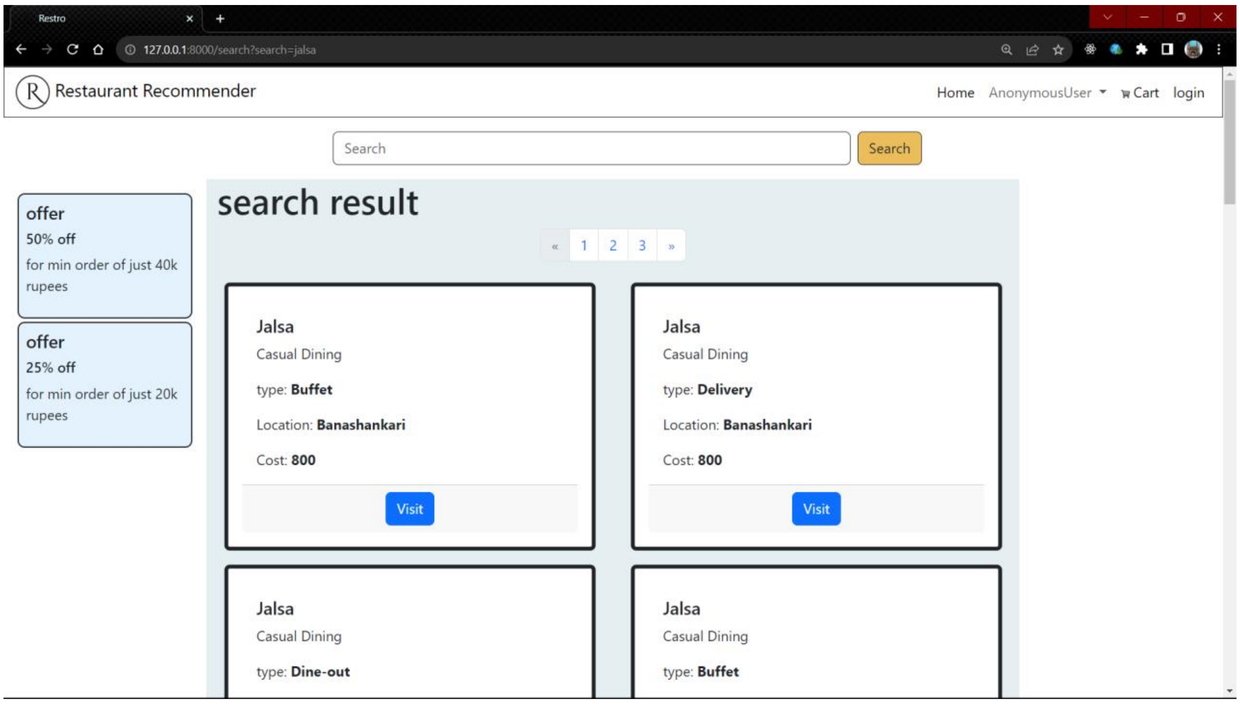
Jalsa
Casual Dining
type: **Buffet**
Location: **Banashankari**
Cost: **800**

Jalsa
Casual Dining
type: **Delivery**
Location: **Banashankari**
Cost: **800**

Jalsa
Casual Dining
type: **Dine-out**

Jalsa
Casual Dining
type: **Buffet**

Figure 8.3 User searching desired restaurant



search result

offer
50% off
for min order of just 40k rupees

offer
25% off
for min order of just 20k rupees

Jalsa
Casual Dining
type: **Buffet**
Location: **Banashankari**
Cost: **800**

Jalsa
Casual Dining
type: **Delivery**
Location: **Banashankari**
Cost: **800**

Jalsa
Casual Dining
type: **Dine-out**

Jalsa
Casual Dining
type: **Buffet**

Fig 8.4 System suggesting the restaurants

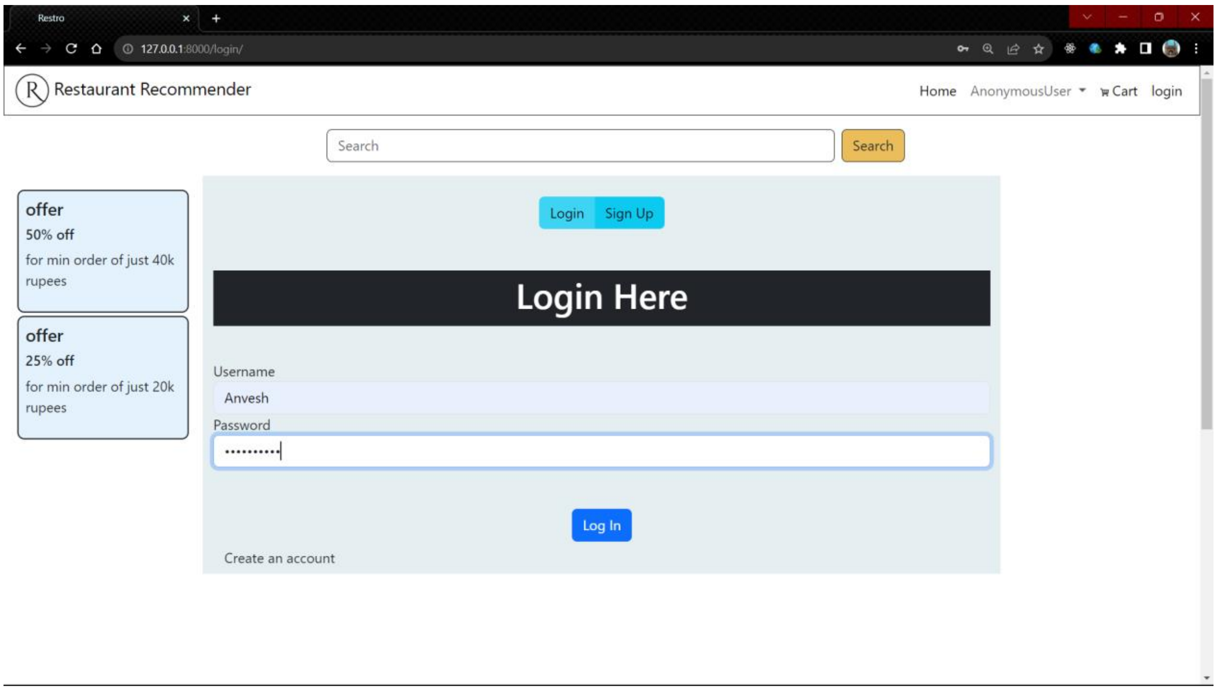


Fig 8.5 System asking to login

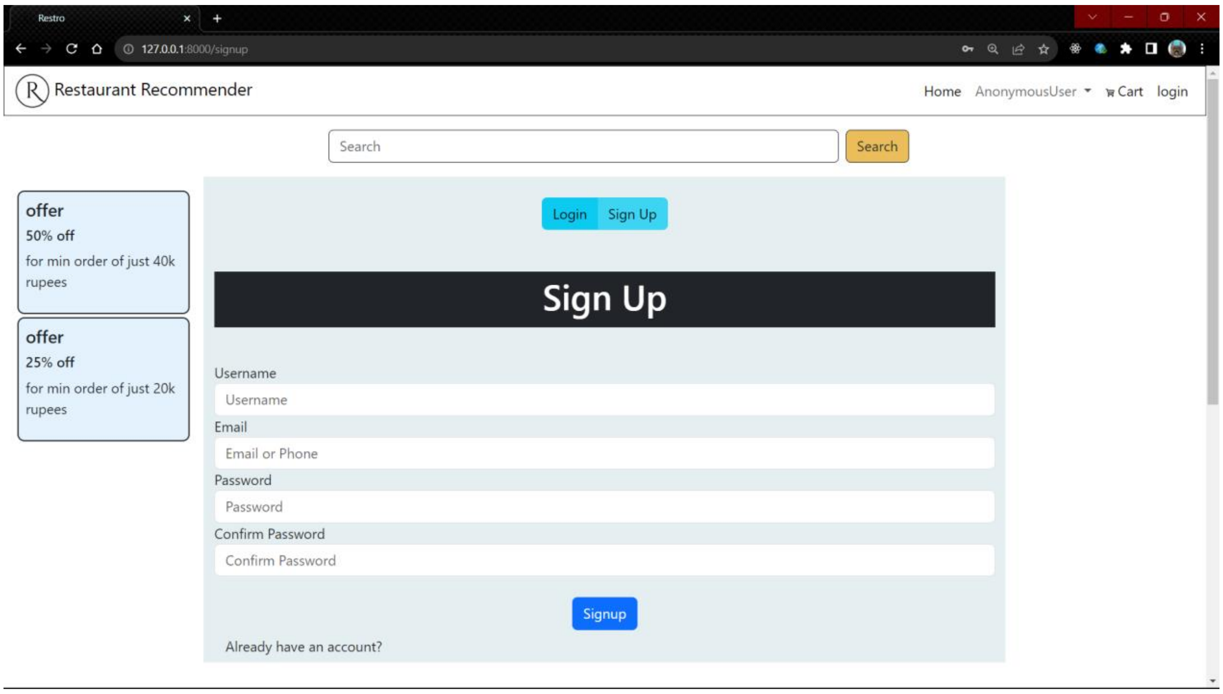
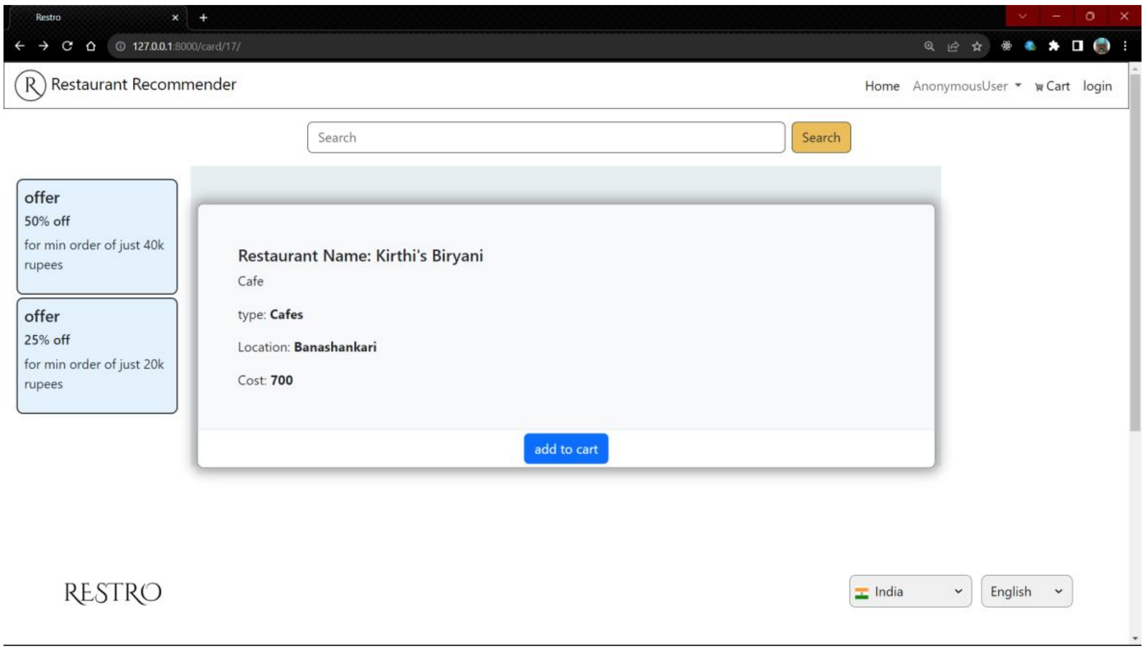


Fig 8.6 System asking to signup



offer
50% off
for min order of just 40k rupees

offer
25% off
for min order of just 20k rupees

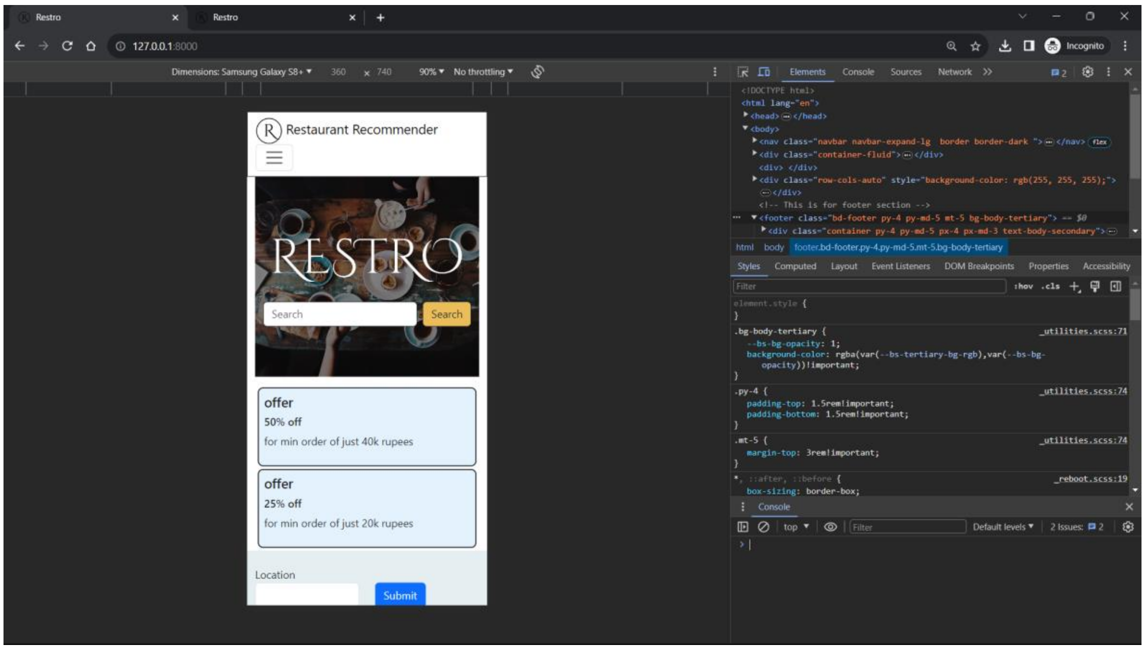
Restaurant Name: Kirthi's Biryani
Cafe

type: **Cafes**

Location: **Banashankari**

Cost: **700**

Fig 8.7 System asking to add in cart



offer
50% off
for min order of just 40k rupees

offer
25% off
for min order of just 20k rupees

```
<!DOCTYPE html>
<html lang="en">
 ▶<head> ⋯ </head>
 ▼<body>
   ▶<nav class="navbar navbar-expand-lg  border border-dark "> ⋯ </nav> flex
   ▶<div class="container-fluid"> ⋯ </div>
    <div> </div>
   ▶<div class="row-cols-auto" style="background-color: rgb(255, 255, 255);">
    ⋯ </div>
    <!-- This is for footer section -->
 ⋯ ▼<footer class="bd-footer py-4 py-md-5 mt-5 bg-body-tertiary"> == $0
     ▶<div class="container py-4 py-md-5 px-4 px-md-3 text-body-secondary"> ⋯
```

html body footer.bd-footer.py-4.py-md-5.mt-5.bg-body-tertiary

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

```
element.style {
}
.bg-body-tertiary {                                        _utilities.scss:71
  --bs-bg-opacity: 1;
  background-color: rgba(var(--bs-tertiary-bg-rgb),var(--bs-bg-
     opacity))!important;
}
.py-4 {                                                    _utilities.scss:74
  padding-top: 1.5rem!important;
  padding-bottom: 1.5rem!important;
}
.mt-5 {                                                    _utilities.scss:74
  margin-top: 3rem!important;
}
*, ::after, ::before {                                        _reboot.scss:19
  box-sizing: border-box;
```

Console

top   Filter   Default levels   2 Issues: 2

Fig 8.8 System is responsive

9. CONCLUSION

In conclusion, the implementation of a restaurant recommendation system using machine learning in real life offers significant benefits for both users and businesses. Leveraging sophisticated algorithms, these systems analyze user preferences, historical data, and contextual information to provide personalized and relevant restaurant suggestions. This not only enhances the user experience by saving time and offering tailored recommendations but also contributes to increased customer satisfaction and loyalty.

For businesses, a well-implemented restaurant recommendation system can boost customer engagement, drive sales, and optimize resource utilization. The system adapts to evolving user preferences, improving its recommendations over time and contributing to a dynamic and customer-centric approach.

Moreover, the real-time nature of these systems ensures that recommendations stay current and aligned with changing trends and user behaviours. As machine learning models continuously learn and evolve based on user interactions, the restaurant recommendation system becomes a valuable tool for staying competitive in the dynamic and ever-changing food and hospitality industry.

In essence, the integration of a machine learning-based restaurant recommendation system in real life reflects a synergy of technological innovation and user-centric design, enriching the dining experience for consumers while providing strategic advantages for businesses in a dynamic and competitive market landscape.

10. FUTURE SCOPE

Looking forward, the future scope for restaurant recommendation systems using machine learning is brimming with possibilities. A deeper dive into personalization could involve the integration of emotion recognition technology, enabling the system to gauge user sentiments in real-time and tailor recommendations based on the emotional context of the dining experience. Exploring the potential of reinforcement learning can introduce an adaptive element to the recommendations, learning from user feedback over time and optimizing suggestions accordingly. Moreover, the incorporation of Bayesian methods could enhance the system's ability to handle uncertainty and provide more nuanced recommendations, especially in situations where user preferences may be ambiguous.

The integration of machine learning interpretability techniques, in addition to explainable AI, can provide users with insights into why specific recommendations are made, fostering a sense of transparency and user control. Moreover, continuous learning models can be fine-tuned with reinforcement learning algorithms, adapting to evolving user preferences dynamically and ensuring that the recommendations remain relevant.

In terms of ethical considerations, the recommendation system could incorporate algorithms to identify and filter out biased or discriminatory content, promoting fair and inclusive suggestions. Additionally, the system could provide information on a restaurant's eco-friendly practices and sourcing methods, aligning with the growing trend of environmentally conscious dining.

The future also holds potential for enhanced user interfaces, with voice-activated commands, chatbot interactions, and virtual assistants further streamlining the user experience. As the Internet of Things (IoT) ecosystem expands, incorporating data from smart kitchen appliances and wearable health devices can contribute additional dimensions to the user profile, enabling even more finely tuned recommendations.

Looking beyond technology, collaborations with nutritionists and health professionals could elevate the system's recommendations to prioritize user health and dietary goals.

Integration with recipe databases and cooking tutorials could extend the user journey beyond restaurant suggestions, empowering users to recreate enjoyable dining experiences at home.

In conclusion, the evolution of restaurant recommendation systems encompasses advancements in interpretability, adaptability, and ethical considerations. By incorporating these aspects and embracing collaborations with diverse stakeholders, the system can evolve into a sophisticated, user-centric platform that not only provides personalized dining suggestions but also contributes positively to user well-being, privacy, and societal values.

REFERENCES

1. Abdulla, G. M., & Borar, S. (2017). Size recommendation system for fashion e-commerce. In KDD Workshop on Machine Learning Meets Fashion.
2. Arora, G., Kumar, A., Devre, G. S., & Ghumare, A. (2014). Movie recommendation system based on users' similarity. International journal of computer science and mobile computing, 3(4), 765-770.
3. Chinsha, T. C., & Joseph, S. (2014). Aspect based opinion mining from restaurant reviews. International Journal of Computer Applications, 975, 8887.
4. Dastidar, S. G. (2017). Restaurant Recommendation System. International Journal of Business Analytics and Intelligence, 5(2), 22.
5. Eyjolfsdottir, E. A., Tilak, G., & Li, N. (2010). Moviegen: A movie recommendation system. UC Santa Barbara: Technical Report.
6. Gandhe, A. (2015). Restaurant recommendation system. cs229. stanford. edu.
7. Jiang, L., Cheng, Y., Yang, L., Li, J., Yan, H., & Wang, X. (2019). A trust-based collaborative filtering algorithm for E-commerce recommendation system. Journal of Ambient Intelligence and Humanized Computing, 10(8), 3023-3034.
8. Jooa, J., Bangb, S., & Parka, G. (2016). Implementation of a recommendation system using association rules and collaborative filtering. Procedia Computer Science, 91, 944-952