Introduction

Our software project is Book Recommender Chatbot. It's a Chatbot application. This application was designed and built on Python for backend and uses HTML, CSS, JS for Frontend. This is a **Rule Based Chatbot**, A **Rule-Based Chatbot uses a tree-like flow instead of AI to help guests with their queries**. This means that the chatbot will guide the guest with follow-up questions to eventually get to the correct resolution. The structures and answers are all pre-defined so that you are in control of the conversation. This Chatbot application can converse in user friendly way and can recommend similar books to the reader based on his interest.

Software Project Description

Motivation of the work

In the present world, all products are buying based on the reviews and ratings by the others. There are so many products with high rating and reviews but we only put our interest in some products which we like. Recommender system works on this principle only i.e., it recommends products based on the interest of the users. Our idea is to create recommender system that recommends books based on the user's interest i.e., we recommend books which are similar to the books that user already liked. It can also recommend books which are liked by similar users. Similar users are those who liked the books which are liked by the current user.

We also add another feature i.e., we recommend books which are independent of the user's interest. With this feature, we can recommend books to the new users also. Book recommendation sites that were available online now a days shows the books which are recommended by the system. Here, we are also recommending books based on the description of the book. We will get books which are similar to the book we selected in this system. For that purpose, we built this system on a combination of Popularity Based Filtering to suggest book to new users, and Collaborating Based Filtering System for recommending book which are similar to the books that user already liked.

Requirements

A requirement is a singular documented physical or functional need that a particular design, product or process aims to safety. It can be divided into functional requirements and non-functional requirements.

Functional Requirement: **-** These are necessary requirement for this Chatbot to be Functional.

- Popularity Based Filtering
- Collaborative Filtering
- Flask Server
- Fetching Response from Server

Non-functional Requirement: -

- Chatbot should be able to give proper recommendation.
- It should be able to properly converse with the user.
- It should not be too intimidating to use for the user.

What is a Recommender System?

Recommender Systems are information filtering systems that help deal with the problem of information overload by filtering and segregating information and creating fragments out of large amounts of dynamically generated information according to user's preferences, interests, or observed behaviour about a particular item or items. A Recommender system has the ability to predict whether a particular user would prefer an item or not based on the user's profile and its Recommender systems. Recommendation systems have also proved to improve the decision-making processes and quality.

In large Recommender systems, recommender systems enhance the revenues for marketing, for the fact that they are effective means of selling more products. In scientific libraries, recommender systems support and allow users to move beyond the generic catalogue searches. Therefore, the need to use efficient and accurate recommendation techniques within a system that provides relevant and dependable recommendations for users cannot be neglected.

Approaches to build Recommender Systems

There are multiple types recommendation engines but we are going to use these two for this project; namely Popularity Based and Collaborative Filtering.

Popularity Based Filtering

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the product or movie which are in trend or are most popular among the users and directly recommend those.

For example, if a product is often purchased by most people, then the system will get to know that that product is most popular so for every new user who just signed it, the system will recommend that product to that user also and chances becomes high that the new user will also purchase that.

Collaborative Filtering

The Collaborative filtering method for recommender systems is a method that is solely based on the past interactions that have been recorded between users and items, in order to produce new recommendations. Collaborative Filtering tends to find what similar users would like and the recommendations to be provided and in order to classify the users into clusters of similar types and recommend each user according to the preference of its cluster. The main idea that governs the collaborative methods is that through past user-item interactions when processed through the system, it becomes sufficient to detect similar users or similar items to make predictions based on these estimated facts and insights.

Tools

Language: Python, HTML, JS, CSS

Packages Used:-

- Flask: Flask is **a micro web framework written in Python**. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
- random: For generating random number.
- NumPy**:** NumPy is **a Python library used for working with arrays**. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

IDE used: **PyCharm** is a dedicated **Python Integrated Development Environment (IDE)** providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Design Level

Package: A python package is a collection of modules. Modules that are related to each other are mainly put in the same package.

Function: In Python, a function is a group of related statements that performs a specific task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes the code reusable.

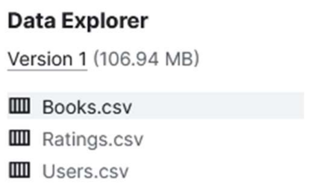
Application Name: Book Recommender System with Chatbot.

Functions:

1. *def recommend(inp)*
2. *def ambiguityResponse()*
3. *def responses(msg, book suggestion)*
4. *def sanitizeText(inp)*
5. *def formatResponse(data, title, author, thumbnail, infoLink)*
6. *def getTopBook()*
7. *def GenerateRecommendation(inp)*
8. *def fetchBooksFromAPI(term)*
9. *def extractProps(items)*

Building Book Recommender Using Chatbot

Firstly, we need data that we are going to use. At Kaggle we have the dataset named “Book Recommendation Dataset”. Here we have three (.csv) file, Data for Books, Users, Ratings. Also, we are going to make a Jupyter Notebook, named “book-recommender-chatbot. We are going to Filter our data for our chatbot from here.



Data Explorer

Version 1 (106.94 MB)

- Books.csv
- Ratings.csv
- Users.csv

Then we are going to import “numpy” and “pandas” libraries for importing and filtering our data.

```
In [1]:

import numpy as np
import pandas as pd
```

Then we are going to use “pandas” library to import our datasets here.

```
In [2]:

books = pd.read_csv('books.csv')
users = pd.read_csv('users.csv')
ratings = pd.read_csv('ratings.csv')
```

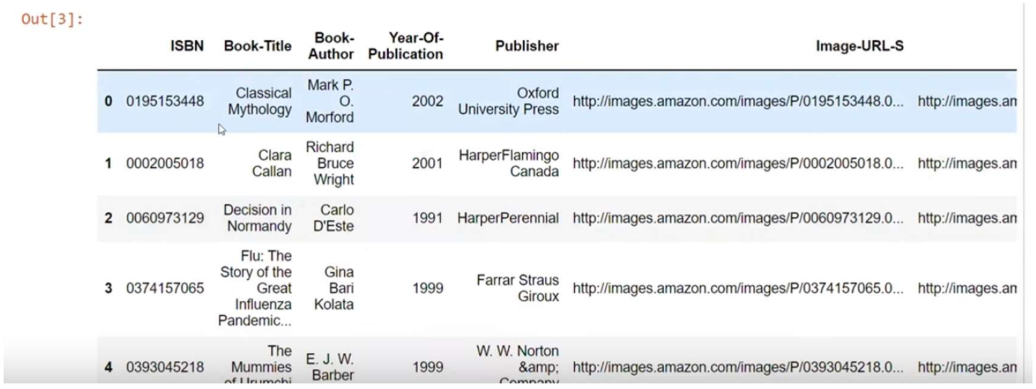
Then we are going to check data set columns.

```
books.head()
```

```
users.head()
```

```
ratings.head()
```

We have the following columns in our Books dataset.



Out[3]:

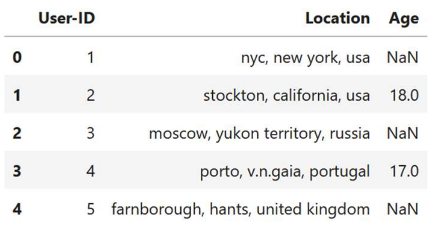
	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.an
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.an
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.an
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.an
4	0393045218	The Mummies	E. J. W. Barber	1999	W. W. Norton &amp;	http://images.amazon.com/images/P/0393045218.0...	http://images.an

So, we have the following column in our books dataset:

ISBN

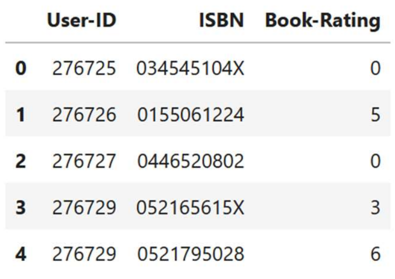
- **Book-Title**
- **Book-Author**
- **Year-Of-Publication**
- **Publisher**
- **Image-URL-S**
- **Image-URL-M**
- **Image-URL-L.**

In Users dataset.



	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

In Ratings dataset.



	User-ID	ISBN	Book-Rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

Let’s see the shape of the data using the shape.

```
print(books.shape)
print(ratings.shape)
print(users.shape)
```

```
(271360, 8)
(1149780, 3)
(278858, 3)
```

Checking for missing data:

```
books.isnull().sum()
```

```
ISBN                   0
Book-Title             0
Book-Author            1
Year-Of-Publication    0
Publisher              2
Image-URL-S            0
Image-URL-M            0
Image-URL-L            3
dtype: int64
```

In Users dataset "Age" column has too many row data missing, but we will not be needing that column so we will ignore that for now.

```
User-ID          0
Location         0
Age         110762
dtype: int64
```

```
ratings.isnull().sum()
```

```
User-ID        0
ISBN           0
Book-Rating    0
dtype: int64
```

Checking for Duplicated Data:

```
books.duplicated().sum()
users.duplicated().sum()
ratings.duplicated().sum()
```

Above code returns "0" which means there is no duplicated data

Popularity Based Recommender System

We are going to filter out Highest Average Rating books, We will display the top Highest Rating Books, and we are going to consider only which have got a minimum of 250 Votes.

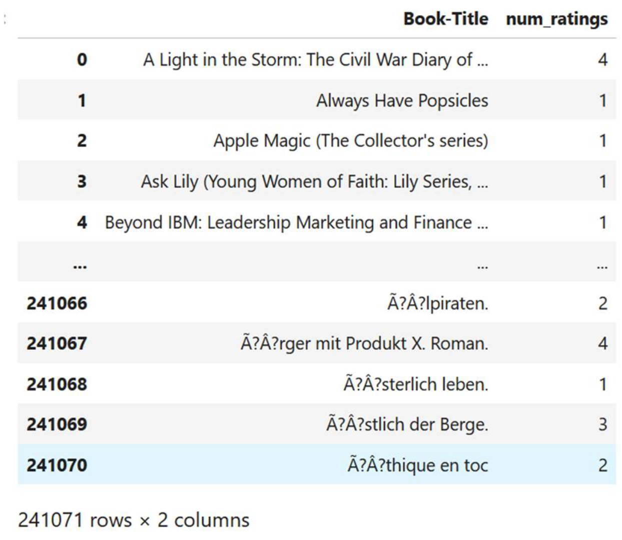
First, we are going to Merge Ratings with Books on top of ISBN columns which are present in both dataframe and store it in a variable.

```
ratings_with_name = ratings.merge(books,on='ISBN')
```

We are going to group data on basis of “Book-Title” to find out no. of votes per books.

```
num_rating_df = ratings_with_name.groupby('Book-Title').count()['Book-Rating
num_rating_df.rename(columns={'Book-Rating':'num_ratings'},inplace=True)
num_rating_df
```

Output:



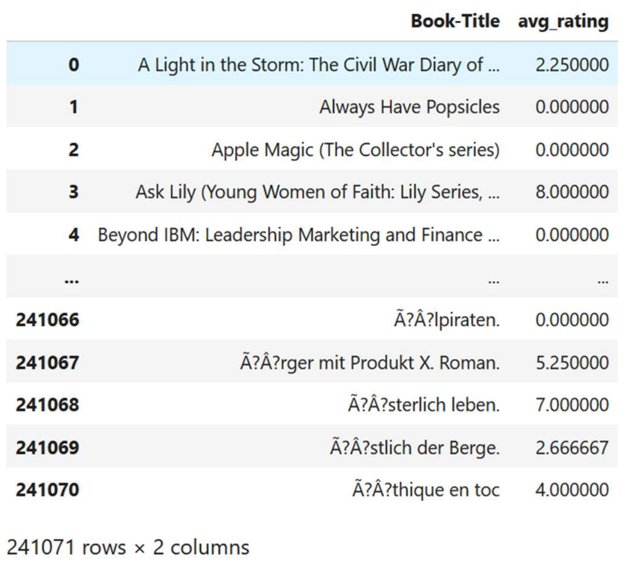
	Book-Title	num_ratings
0	A Light in the Storm: The Civil War Diary of ...	4
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Ask Lily (Young Women of Faith: Lily Series, ...	1
4	Beyond IBM: Leadership Marketing and Finance ...	1
...	...	...
241066	Ã?Â?lpiraten.	2
241067	Ã?Â?rger mit Produkt X. Roman.	4
241068	Ã?Â?sterlich leben.	1
241069	Ã?Â?stlich der Berge.	3
241070	Ã?Â?thique en toc	2

241071 rows × 2 columns

Then we are going to find average rating on each books:

```
avg_rating_df = ratings_with_name.groupby('Book-Title').mean()['Book-Rating'].reset_index()
avg_rating_df.rename(columns={'Book-Rating':'avg_rating'},inplace=True)
avg_rating_df
```

Output:



	Book-Title	avg_rating
0	A Light in the Storm: The Civil War Diary of ...	2.250000
1	Always Have Popsicles	0.000000
2	Apple Magic (The Collector's series)	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	0.000000
...	...	...
241066	Ã?Â?lpiraten.	0.000000
241067	Ã?Â?rger mit Produkt X. Roman.	5.250000
241068	Ã?Â?sterlich leben.	7.000000
241069	Ã?Â?stlich der Berge.	2.666667
241070	Ã?Â?thique en toc	4.000000

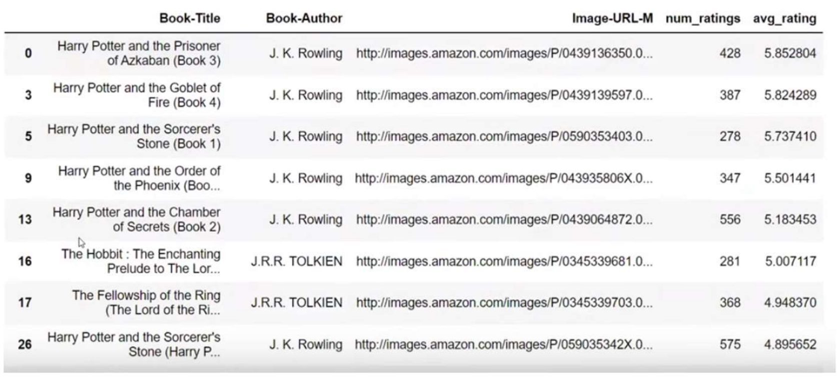
241071 rows × 2 columns

Then we are going to merge these dataframe with each other on basis of “Book-Title”

```
popular_df = popular_df[popular_df['num_ratings']>=250].sort_values('avg_rating',ascending=False).head(50)
```

```
popular_df = popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Title','Book-Author','Image-URL-M','num_ratings','avg_rating']]
```

Resulting Dataframe:



	Book-Title	Book-Author	Image-URL-M	num_ratings	avg_rating
0	Harry Potter and the Prisoner of Azkaban (Book 3)	J. K. Rowling	http://images.amazon.com/images/P/0439136350.0...	428	5.852804
3	Harry Potter and the Goblet of Fire (Book 4)	J. K. Rowling	http://images.amazon.com/images/P/0439139597.0...	387	5.824289
5	Harry Potter and the Sorcerer's Stone (Book 1)	J. K. Rowling	http://images.amazon.com/images/P/0590353403.0...	278	5.737410
9	Harry Potter and the Order of the Phoenix (Boo...	J. K. Rowling	http://images.amazon.com/images/P/043935806X.0...	347	5.501441
13	Harry Potter and the Chamber of Secrets (Book 2)	J. K. Rowling	http://images.amazon.com/images/P/0439064872.0...	556	5.183453
16	The Hobbit : The Enchanting Prelude to The Lor...	J.R.R. TOLKIEN	http://images.amazon.com/images/P/0345339681.0...	281	5.007117
17	The Fellowship of the Ring (The Lord of the Ri...	J.R.R. TOLKIEN	http://images.amazon.com/images/P/0345339703.0...	368	4.948370
26	Harry Potter and the Sorcerer's Stone (Harry P...	J. K. Rowling	http://images.amazon.com/images/P/059035342X.0...	575	4.895652

Thus, we have a dataframe of Top Books which have minimum of 250 Votes.

Collaborative Filtering Based Recommender System

We are going to use the "ratings_with_name" which we used previously for popularity based dataframe here again

```
ratings_with_name = ratings.merge(books,on='ISBN')
```

We are going to use "groupby()" on the basis for "User-ID" and select only the user which had rated 200 books at minimum we are going to call them "experienced_user" as they have a good reading experience. We found out this parameter after researching the data.

```
x = ratings_with_name.groupby('User-ID').count()['Book-Rating'] > 200
experienced_user = x[x].index
filtered_rating = ratings_with_name[ratings_with_name['User-ID'].isin(experienced_user)]
```

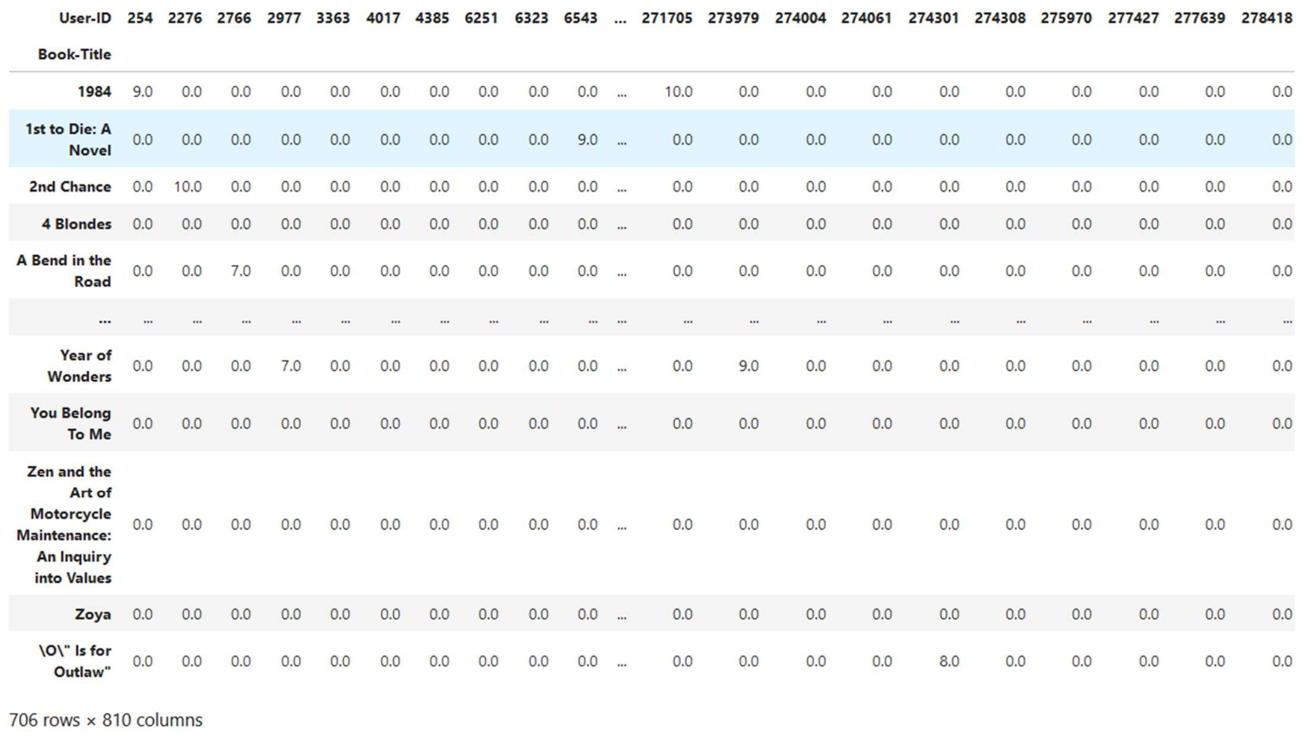
We have filtered on the basis of users, now we are going to filter on the basis of books.

```
y = filtered_rating.groupby('Book-Title').count()['Book-Rating']>=50
famous_books = y[y].index
```

Now we are going to make a pivot table on this data set "final_ratings".

```
final_ratings = filtered_rating[filtered_rating['Book-Title'].isin(famous_books)]
pt = final_ratings.pivot_table(index='Book-Title',columns='User-ID',values='Book-Rating')
pt.fillna(0,inplace=True)
pt
```

This is the resulting Pivot Table:



User-ID	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	271705	273979	274004	274061	274301	274308	275970	277427	277639	278418
Book-Title																					
1984	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2nd Chance	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4 Blondes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A Bend in the Road	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Year of Wonders	0.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
You Belong To Me	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Zoya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
\O\" Is for Outlaw"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0

706 rows × 810 columns

Now assume if this pivot table is 810th dimensional space then the Book-Title are vectors in that space. We are going to find similarity between these books by comparing and finding the closest books distance with the searched book to find the similarity in them.

We will import “cosine_similarity” and find cosine similarity of all the rows with all the rows.

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_scores = cosine_similarity(pt)
similarity_scores.shape
```

Result:

```
(706, 706)
```

Now we need to write a function for named “recommend()” which will take input of a book name and return a suggestion of five books.

```python
def recommend(book_name):
    # index fetch
    index = np.where(pt.index==book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))

        data.append(item)

    return data
```

On running this function, the output:

```python
recommend('1984')
```

```
[['Animal Farm',
  'George Orwell',
  'http://images.amazon.com/images/P/0451526341.01.MZZZZZZZ.jpg'],
 ["The Handmaid's Tale",
  'Margaret Atwood',
  'http://images.amazon.com/images/P/0449212602.01.MZZZZZZZ.jpg'],
 ['Brave New World',
  'Aldous Huxley',
  'http://images.amazon.com/images/P/0060809833.01.MZZZZZZZ.jpg'],
 ['The Vampire Lestat (Vampire Chronicles, Book II)',
  'ANNE RICE',
  'http://images.amazon.com/images/P/0345313860.01.MZZZZZZZ.jpg']]
```

As you can see the function outputs an array of 5 books details which are suggestions.

Now we can export this dataframe to use it in our chatbot application. For that we need to use “Pickle” libraries.

```python
import pickle
pickle.dump(popular_df,open('popular.pkl','wb'))
```

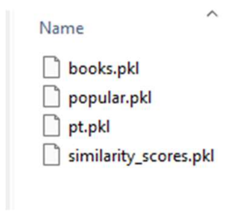
Removing duplicates:

```python
books.drop_duplicates('Book-Title')
```

Exporting Other data frame:

```python
pickle.dump(pt,open('pt.pkl','wb'))
pickle.dump(books,open('books.pkl','wb'))
pickle.dump(similarity_scores,open('similarity_scores.pkl','wb'))
```

By doing this we get “.pkl” files of the exported data frame


Name
books.pkl
popular.pkl
pt.pkl
similarity_scores.pkl

The Chatbot.

We are going to make a rule-based chatbot. For that we will have to follow the following steps:

1. Importing libraries

For the project, we are importing the necessary modules. We will use flask library for creating the web interface for the chatbot, pickle to unpack and use the data as dictionary, NumPy to handle multidimensional arrays

```
import pickle
import random
import re

import numpy as np
import requests
from flask import Flask, render_template, request
```

2. Loading Pickle Files

Now, we need to unpack data that that is required for the system to function using pickle. These files were generated exported using pickle library in Jupyter Notebook after applying Filtering Techniques.

```
# loading pickle files
popular_df = pickle.load(open('popular.pkl', 'rb'))
pt = pickle.load(open('pt.pkl', 'rb'))
books = pickle.load(open('books.pkl', 'rb'))
similarity_scores = pickle.load(open('similarity_scores.pkl', 'rb'))
```

Defining our recommend() function:

Now that we have imported our data frames from Jupyter Notebook, we will define our recommend() function that we made in Jupyter for recommending similar books.

```
# For finding similar recommendation :used in getResponse():
def recommend(inp):
    if inp == "":
        return "Please Enter a book name."
    else:
        index = np.where(pt.index == inp)[0][0]
        similar_items = sorted(
            list(enumerate(similarity_scores[index])), key=lambda x: x[1],
reverse=True)[1:5]
        data = []
        for i in similar_items:
            item = []
            temp_df = books[books['Book-Title'] == pt.index[i[0]]]
            item.extend(list(temp_df.drop_duplicates(
                'Book-Title')['Book-Title'].values))
            item.extend(list(temp_df.drop_duplicates(
                'Book-Title')['Book-Author'].values))
            item.extend(list(temp_df.drop_duplicates(
                'Book-Title')['Image-URL-M'].values))

            data.append(item)
            strr = "<h3>Here are some similar books </h3></br> \n"
        for book in data:
            strr += """
                        <p><b>Book Name: </b>""" + book[0]+""" </p><br/>
                        <p><b>Author: </b>"""+book[1].capitalize() + """</p>
                        <br/><a  target='_blank' href='https://www.google.com/search?q="""
+ book[0] + """ book'><img src=' """ + book[2] + """ ' ></a><br/><br/>
                    """
        return strr
```

This function will take the input as “inp” from other function that sanitizes the input and will search for the term in the dataset, and return books detail formatted with html elements that will be required later for showing in chatbot thread.

1. The “responses()” function:

Now we define a function which will be required for our chatbot to be able to understand give reasonable reply to the user. We will name it responses () function. This function contains keywords and their responses which will be chosen based on the user input, there is also a token keyword named “recommend:” which will regard as a trigger and input for our recommend() function.

```
# This contain list of rules and responses.:Used in GetTopBook():
def responses(msg, book_suggestion):
    inp = sanitizeText(msg)

    if inp == "hello":
        return "Hi there, how can I help? 😊"
    elif inp == "hi" or inp == "hii" or inp == "hiiii":
        return "Hi there, what can I do for you?🙂"
    elif inp == "how are you" or inp == "how are you?":
        return "Fine! and you?"
    elif inp == "fine" or inp == "i am fine" or inp == "i m fine" or inp == "i am good" or
inp == "i am doing good":
        return "Great! how can I help you."
    elif inp == "thanks" or inp == "thank you" or inp == "now its my time":
        return "My pleasure !👌😄"
    elif inp == "what do you do" or inp == "have you something":
        return "Well I can give you recommendation."
    elif inp == "tell me a joke" or inp == "tell me something funny" or inp == "crack a
funny line":
        return "What did the buffalo say when his son left for college? Bison!"
    elif inp == "okay" or inp == "hmm" or inp == "yup":
        return "Okay"
    elif inp == "ok" or inp == "okk" or inp == "okkkkk":
        return "Hmm"
    elif inp == "goodbye" or inp == "good bye" or inp == "bye" or inp == "see you later"
or inp == "see yaa":
        return "Have a nice day!"
    elif inp == "show me a recommendation" or inp == "show me a book" or inp == "give me a
book to read":
        return "<p> Here you go! 🙂 " + book_suggestion + "</p>"
    elif inp == "i am bored" or inp == "im bored" or inp == "i m bored" or inp == "i'm
bored":
        return "<p> Read this book 🙂 " + book_suggestion + "</p>"
    elif inp == "recommend me a book" or inp == "suggest me a book" or inp == "suggest me
something interesting":
        return "<p> I hope you like it!😊 " + book_suggestion + "</p>"
    elif inp == "what should i read today" or inp == "recommend me something":
        return "<p> This one seems interesting enough! " + book_suggestion + "</p>"
    else:
        return ambiguityResponse()
```

2. The “ambiguityResponse()” function:

This function returns random responses as if it didn't understand. It is used in responses() function.

```
# Output random responses as if it didn't understand :Used in responses():

def ambiguityResponse():
    rand = random.randint(0, 5)
    response = [
        "Sorry! I didn't understand that😥😓",
        "I beg your pardon?", "Sorry, I’m afraid I don’t follow you.",
        "Excuse me, could you repeat the question?",
        "I’m sorry, I don’t understand.",
        "I’m confused. Could you tell me again?",
        "I don’t get it…"
    ]
    return response[rand]
```

3. The “sanitizeText()” function:

This function will sanitize the user input and prepare the text for passing to the function.

```
# This function will sanitize the user input and prepare the text for passing to the
function :Used in responses():
def sanitizeText(inp):
    # removing extra spaces
    " ".join(inp.split())

    # Converting to Lower
    inp = inp.lower()

    # Removing unnecessary symbols & punctuation
    inp = re.sub('[^A-Za-z0-9 ]+', '', inp)
    return inp
```

4. The “getTopBook()” function:

This function returns a random book

```
# for getting top books :Used in GenerateRecommendation():
def getTopBook():
    # Picking a Random Book from Top Selling books
    book_name = list(popular_df['Book-Title'].values)
    book_author = list(popular_df['Book-Author'].values)
    img_url = list(popular_df['Image-URL-M'].values)
    rand = random.randint(0, 49)
    book = book_name[rand]
    author = book_author[rand]
    img = img_url[rand]
    data = {"book": book, "img": img, "author": author}
    return data
```

The “GenerateRecommendation()” function:

This function will function will return a random book recommendation with proper html encasing. This output is then passed to responses() along with the user input.

```
# This function will return a random book recommendation.
def GenerateRecommendation(inp):
    TopBook = getTopBook()
    infoLink = "https://www.google.com/search?q=" + TopBook['book'] + " book"
    book_suggestion = formatResponse(
        TopBook, "book", "author", "img", infoLink)

    return responses(inp, book_suggestion)
```

The “formatResponse ()” function:

This function will return a books In html format which will be then sent as response to the client browser.

```
def formatResponse(data, title, author, thumbnail, infoLink):
    book_suggestion = """
    <div class='book'>
        <a style='margin:5px' target='_blank' href='"""+infoLink+"""'><img src='""" +
data[thumbnail] + """' /></a>
        <p><b>Book Name: </b>""" + data[title] + """ </p>
        <p><b>Author: </b>""" + data[author] + """</p>
    </div>
    """
    return book_suggestion
```

The “fetchBooksFromAPI ()” function:

This function will ping the Google Books API to fetch necessary information on the queried books(eg:- Image Thumbnail, Info Links, Author Name etc.) and return it in JSON (JavaScript Object Notation) format, which we can use to extract necessary fields that we need.(This request Doesn’t require an API Key).

```
# fetch results from google books api
def fetchBooksFromAPI(term):
    bookUrl = "https://www.googleapis.com/books/v1/volumes"
    apiKey = "key=AIzaSyDtXC7kb6a7xKJdm_Le6_BYoY5biz6s8Lw"
    placeHldr = '<img src="https://via.placeholder.com/150">'
    # Here wil come the keyword that user is searching for from (User Input)
    searchData = ""
    searchData = {"q": term}
    if searchData == "" or searchData == None:
        print("Search Term cannot be blank")
    else:
        x = requests.get(bookUrl, params=searchData, headers={
                         'Accept': 'application/json'})
        data = x.json()

    return data
```

The “extractProps ()” function:

This function will extract and return necessary columns for each books and make a list a set of list.

```
def extractProps(items):
    rows = []
    for r in items:
        if 'title' not in r['volumeInfo'] or 'authors' not in r['volumeInfo'] or
'subtitle' not in r['volumeInfo'] or 'imageLinks' not in r["volumeInfo"]:
            continue

        title = r['volumeInfo']['title']
        subtitle = r['volumeInfo']['subtitle']
        infoLink = r['volumeInfo']['infoLink']
        author = r['volumeInfo']['authors'][0]
        thumbnail = r['volumeInfo']['imageLinks']['thumbnail']
        rows += [dict(title=title, subtitle=subtitle, author=author,
                      thumbnail=thumbnail, infoLink=infoLink)]
    return rows
```

Coding The Frontend

Now that our backend is completed we will code the frontend. We will be using the flask library to code the frontend in HTML, CSS, & JS.
For that we will need to create a “templates” folder in the root directory and create a “chats.html” file and make a Simple chat user interface.

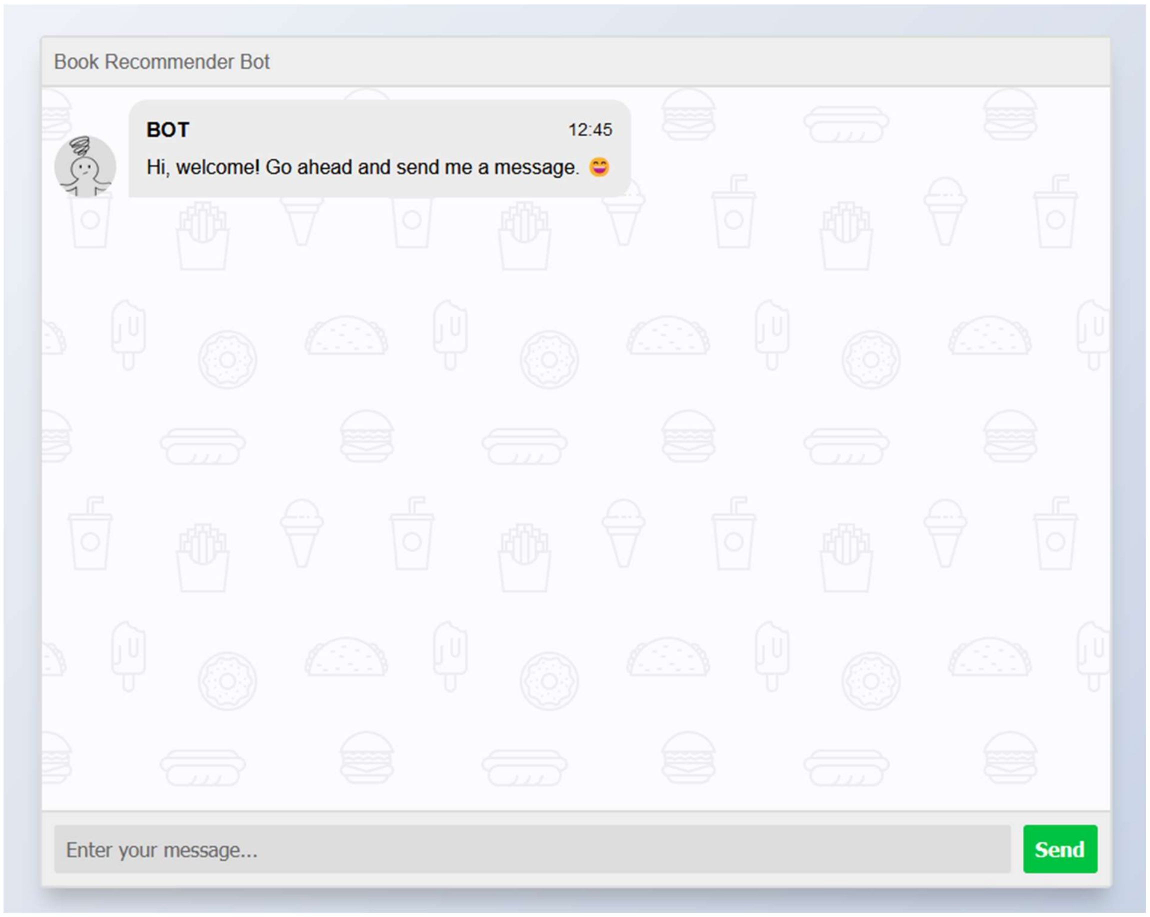
index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Chat</title>
    <link rel="stylesheet" href="{{url_for('static', filename='app.css')}}">
</head>
<body>
    <section class="msger">
      <header class="msger-header">
        <div class="msger-header-title">
          <i class="fas fa-comment-alt"></i> Book Recommender Bot
        </div>
        <div class="msger-header-options">
          <span><i class="fas fa-cog"></i></span>
        </div>
      </header>

      <main class="msger-chat">
        <div class="msg left-msg">
          <div
           class="msg-img"
           style="background-image: url(https://cdn-icons-
png.flaticon.com/512/6684/6684500.png)"
          ></div>

          <div class="msg-bubble">
            <div class="msg-info">
              <div class="msg-info-name">BOT</div>
              <div class="msg-info-time">12:45</div>
            </div>

            <div class="msg-text">
              Hi, welcome! Go ahead and send me a message. 😄
            </div>
          </div>
        </div>

        </div>
      </main>
      <form class="msger-inputarea">
        <input type="text" class="msger-input" placeholder="Enter your message...">
        <button type="submit" class="msger-send-btn">Send</button>
      </form>
    </section>
</body>
```

```
<script src="{{url_for('static', filename='app.js')}}"></script>
</html>
```

HTML Preview

After applying some necessary CSS Styling to our index.html file it should look like this.



app.js

Here is a JavaScript function to get input from user and forward it to the server:

```
function getResponseFromServerBot(message){
    let msg = "This is Message";
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function(msg) {
        msg =  this.responseText;
        botResponse(msg);
    }
  xhttp.open("GET", "/getResponse?msg="+message, true);
  xhttp.send();

}
```

This function is for appending the response to the thread

```
function appendMessage(name, img, side, text) {
  //   Simple solution for small apps
  const msgHTML = `
    <div class="msg ${side}-msg">
      <div class="msg-img" style="background-image: url(${img})"></div>

      <div class="msg-bubble">
        <div class="msg-info">
          <div class="msg-info-name">${name}</div>
          <div class="msg-info-time">${formatDate(new Date())}</div>
        </div>

        <div class="msg-text">${text}</div>
      </div>
    </div>
  `;

  msgerChat.insertAdjacentHTML("beforeend", msgHTML);
  msgerChat.scrollTop += 500;
}
```

This function is to set bot side of response.

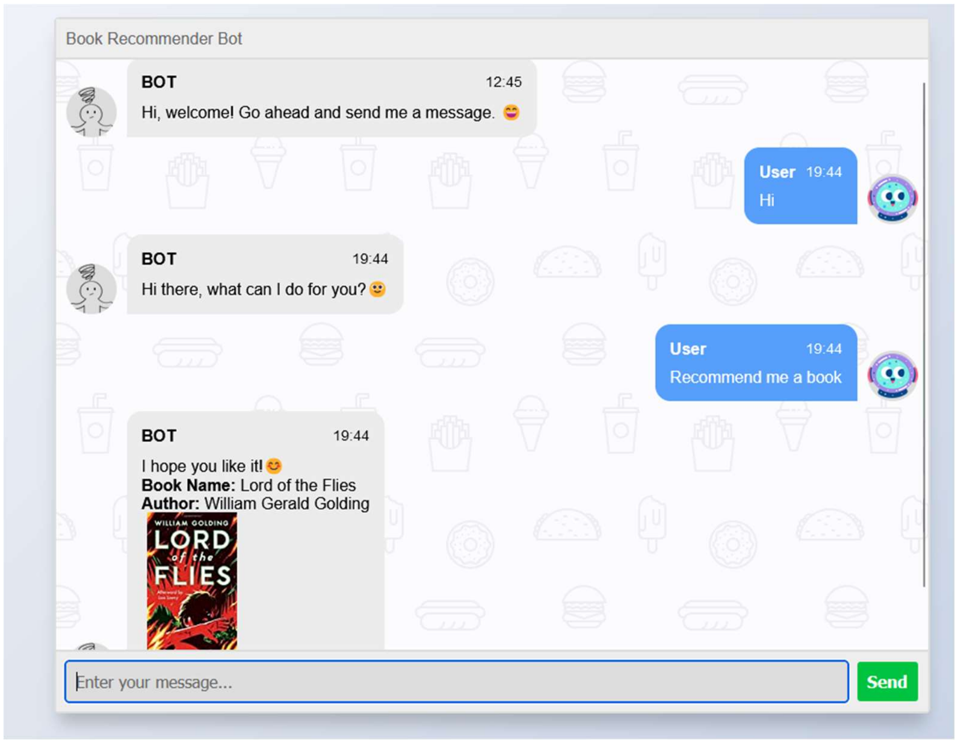
```
function botResponse(botReply) {
    const delay = botReply.split(" ").length * 100;
//    log(botReply)
    setTimeout(() => {
        appendMessage(BOT_NAME, BOT_IMG, "left", botReply); //This appends the message to
bot side
    }, 2000);
}
```

For appending User message to the thread and sending User message to the Server.

```
msgerForm.addEventListener("submit", event => {
    event.preventDefault();
    const msgText = msgerInput.value;
    if (!msgText) return;

    //Appending the message
    appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
    msgerInput.value = "";
    botReply = getResponseFromServerBot(msgText);

});
```

Preview

Random Top-Book Recommendation:



Book Recommender Bot

BOT 12:45
Hi, welcome! Go ahead and send me a message. 😄

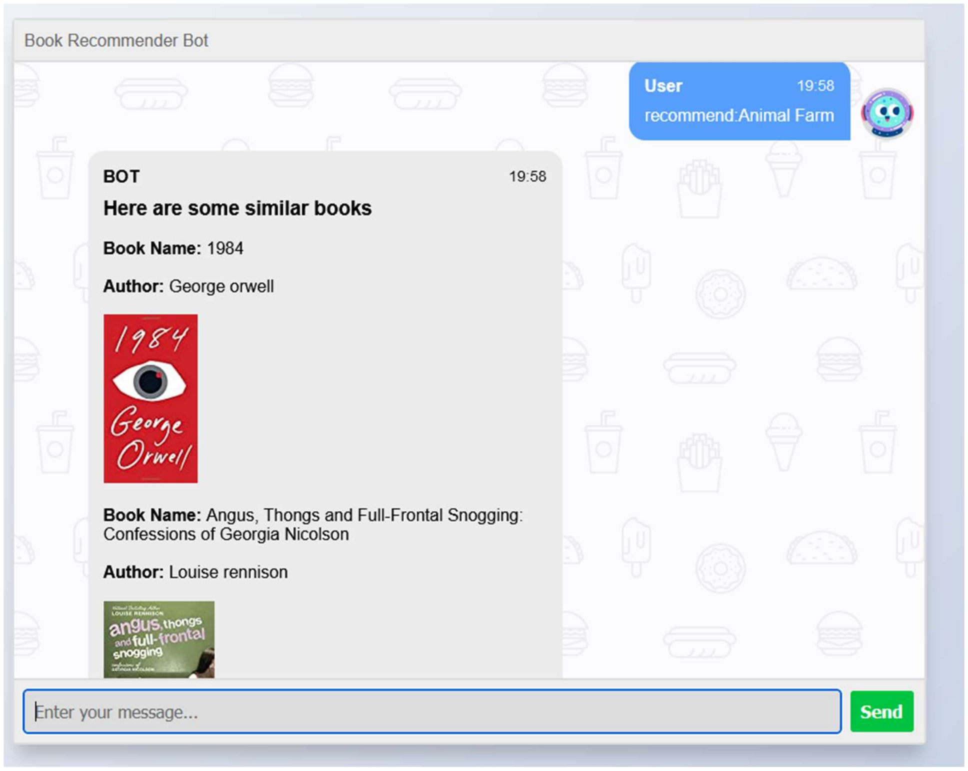
User 19:44
Hi

BOT 19:44
Hi there, what can I do for you? 🙂

User 19:44
Recommend me a book

BOT 19:44
I hope you like it! 😊
Book Name: Lord of the Flies
Author: William Gerald Golding

The Similar Book Recommender System:



Book Recommender Bot

User 19:58
recommend:Animal Farm

BOT 19:58
Here are some similar books

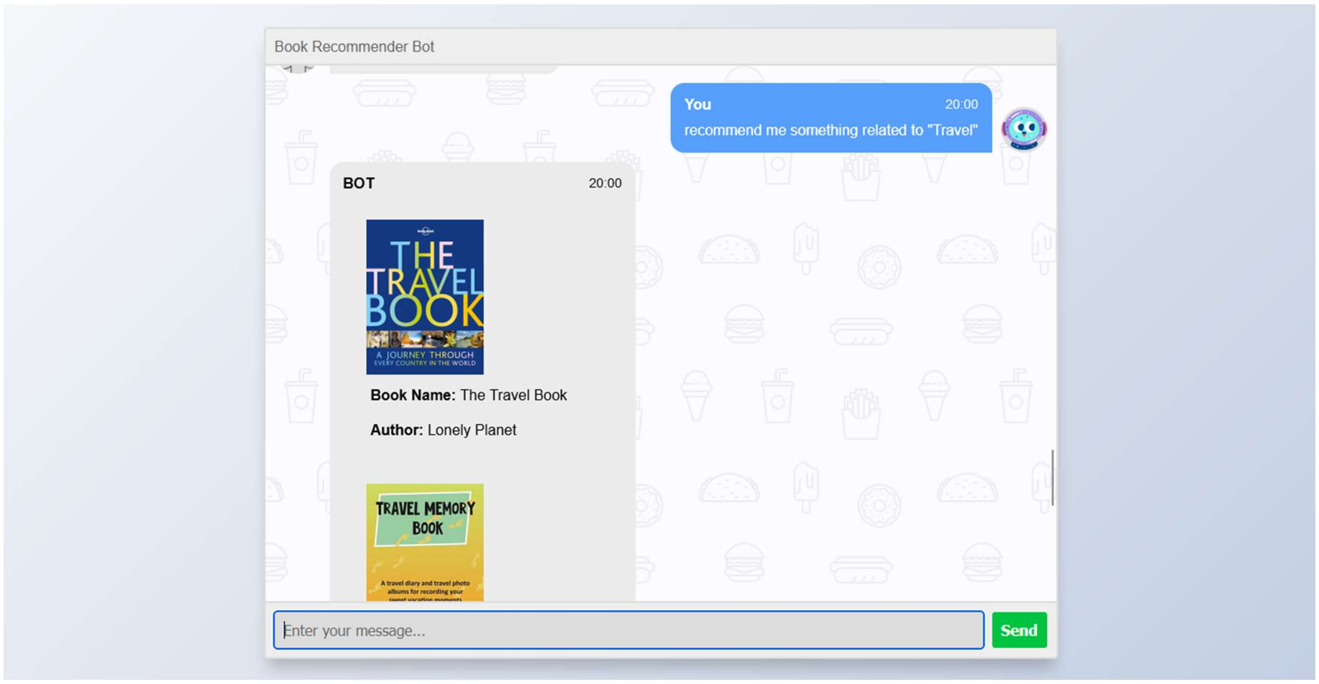
Book Name: 1984

Author: George orwell

Book Name: Angus, Thongs and Full-Frontal Snogging: Confessions of Georgia Nicolson

Author: Louise rennison

The Genre wise Searching System:



Book Recommender Bot

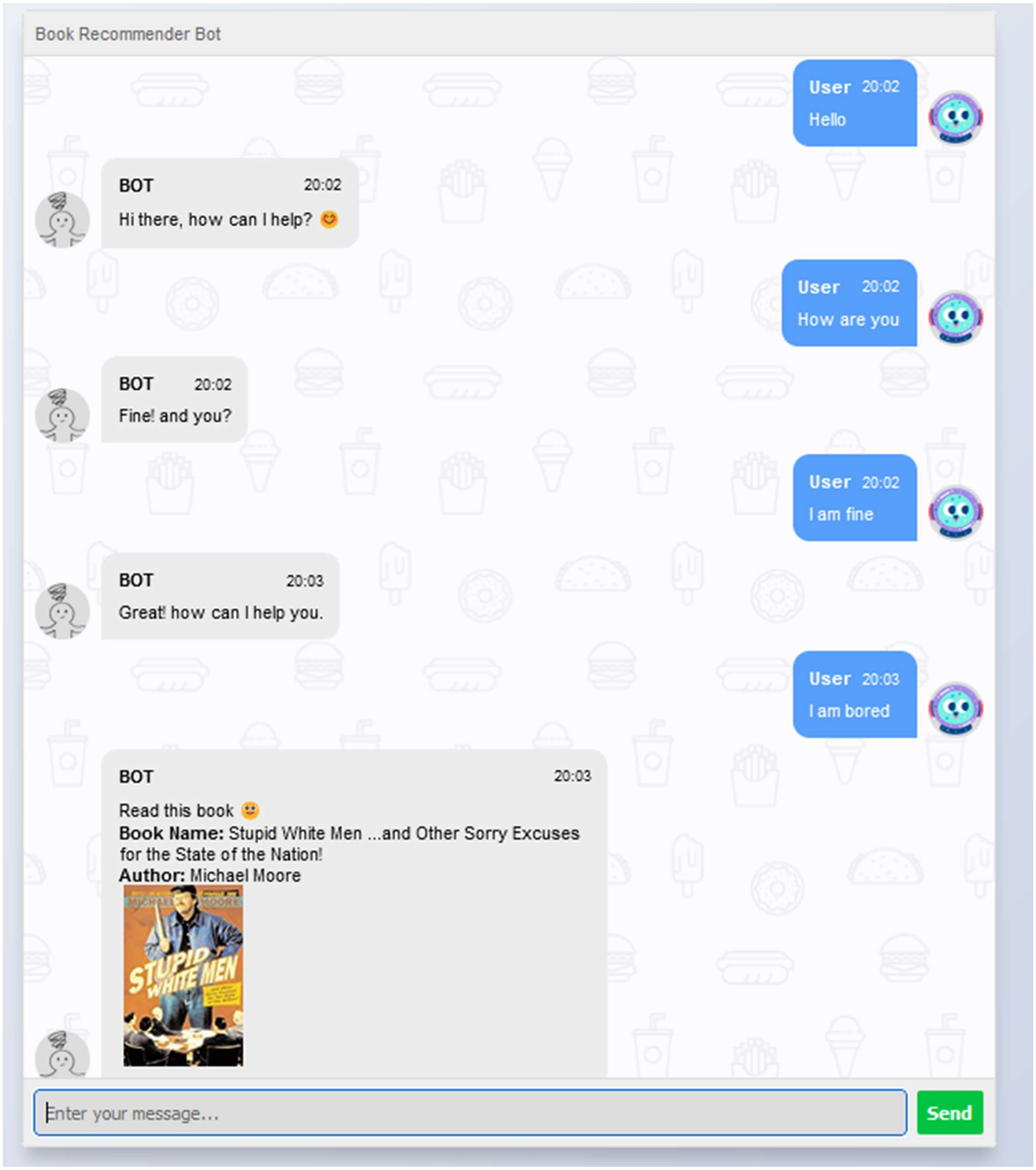
You 20:00
recommend me something related to "Travel"

BOT 20:00

Book Name: The Travel Book

Author: Lonely Planet

Normal Conversations:



Book Recommender Bot

User 20:02
Hello

BOT 20:02
Hi there, how can I help? 😊

User 20:02
How are you

BOT 20:02
Fine! and you?

User 20:02
I am fine

BOT 20:03
Great! how can I help you.

User 20:03
I am bored

BOT 20:03
Read this book 🙂
Book Name: Stupid White Men ...and Other Sorry Excuses for the State of the Nation!
Author: Michael Moore

Software Project Deliverables

1. Project report
2. Source Code

Summary

We choice this project to gain proper knowledge to make Web applications using Python. It increased our knowledge for Python development. Getting experience in different Python packages.

Reference

1. The Dataset used in the Project
2. Python NumPy
3. Pandas Data Frame
4. Flask Tutorial
5. Icons used in this project