App.py

```
# import the librareis

import streamlit as st

import pickle

import numpy as np

import pandas as pd

st.set_page_config(layout="wide")

st.header("Book Recommender System")

st.markdown('''

        ##### The site usinging colaborative filtering suggests books from our
catalog.

        ##### We recommend top 50 books for every one as well.

        ''')

# import our models :

popular = pickle.load(open('popular.pkl','rb'))

books = pickle.load(open('books.pkl','rb'))

pt = pickle.load(open('pt.pkl','rb'))
```

```
similarity_scores = pickle.load(open('similarity_scores.pkl','rb'))

# Top 50 Books :

st.sidebar.title("Top 50 Books")

if st.sidebar.button("SHOW"):

    cols_per_row = 5

    num_rows = 10

    for row in range(num_rows):

        cols = st.columns(cols_per_row)

        for col in range(cols_per_row):

            book_idx = row * cols_per_row + col

            if book_idx < len(popular):

                with cols[col]:

                    st.image(popular.iloc[book_idx]['Image-URL-M']) # Displays the
image

                    st.text(popular.iloc[book_idx]['Book-Title']) # Displays the Book
Title

                    st.text(popular.iloc[book_idx]['Book-Author']) # Display the Author
name

# Function to recommed Books
```

```
def recommend(book_name):

    index = np.where(pt.index == book_name)[0][0]

    similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda
x : x[1], reverse=True)[1:6]

    # Lets create empty list and in that lies i want ot populate with the book
information

    # Book author book-title image url

    # Empty list

    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

# this is giving the names list of books.

book_list = pt.index.values

st.sidebar.title("Similar Book Suggestions")

# Dro down to select the books

selected_book = st.sidebar.selectbox("Select a book from the dropdown",
book_list)

if st.sidebar.button("Recommend Me"):

    book_recommend = recommend(selected_book)

    cols = st.columns(5)

    for col_idx in range(5):

        with cols[col_idx]:

            if col_idx < len(book_recommend):

                st.image(book_recommend[col_idx][2])

                st.text(book_recommend[col_idx][0])

                st.text(book_recommend[col_idx][1])

# import data

# books = pd.read_csv('Books.csv')  # books data
```

```
books = pd.read_csv('Books.csv', low_memory=False)

users = pd.read_csv('Users.csv') # Users location and age data

ratings = pd.read_csv('Ratings.csv') # Users rating data

st.sidebar.title("Data Used")

if st.sidebar.button("Show"):

    st.subheader('This is the books data we used in our model')

    st.dataframe(books)

    st.subheader('This is the User ratings data we used in our model')

    st.dataframe(ratings)

    st.subheader('This is the user data we used in our model')
    st.dataframe(users)
```

Recommender1.ipynb

```
import numpy as np

import pandas as pd

from sklearn.metrics.pairwise import cosine_similarity

# to ignore warinings
```

```
import warnings

warnings.filterwarnings('ignore')

# import data

books = pd.read_csv('Books.csv')  # books data

users = pd.read_csv('Users.csv') # Users location and age data

ratings = pd.read_csv('Ratings.csv') # Users rating data

books.head()

users.head()

ratings.head()

books.shape

ratings.shape

users.shape

# Looking of nulls in books data

books.isnull().sum()

# Brop the nulls

books = books.dropna()

# Looking of nulls in books data
```

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

users.isnull().sum()

users = users.dropna()

# Looking of nulls in books data

users.isnull().sum()

# Looking of nulls in books data

ratings.isnull().sum()

books.shape

users.shape

ratings.shape

# Checking of duplicates

books.duplicated().sum()

# Checking of duplicates

users.duplicated().sum()

# Checking of duplicates

ratings.duplicated().sum()

# Unique count
```

```
books.nunique()

users.head()

ratings.head()

np.sort(ratings['Book-Rating'].unique())

books.info()

books.columns

# convert year of publication to int

books['Year-Of-Publication'] = books['Year-Of-Publication'].astype('int32')

books.info()

# Joining books and user ratings into one table

books_with_ratings = ratings.merge(books, on = 'ISBN')

popular_df = popular_df.reset_index()

popular_df.sort_values('num_rating',ascending=False)

# Popularity is based on the no of people read the book  ('num_raitng' > 300)

# It is based on the rating it got.

popular_df =
popular_df[popular_df['num_rating']>300].sort_values('avg_rating',
ascending=False)
```

```
popular_df = popular_df.head(50)

# For the model deployment I need Book-title, Author, Image URL

popular_df = popular_df.merge(books, on = 'Book-
Title').drop_duplicates('Book-Title')[['Book-Title',

                                                  'Book-Author',

                                                  'Image-URL-M',

                                                  'num_rating',

                                                 'avg_rating']]

# Grouping based on user-id tells the no of books rated by each user:

x = books_with_ratings.groupby('User-ID').count()

x

# Select only users who atleast gave feed back for 200 books (Power users )

x = x['Book-Rating'] > 200

x

power_users = x[x].index

power_users.sort_values()

# selecting only records of power users

filtered_ratings = books_with_ratings[books_with_ratings['User-
ID'].isin(power_users)]

# I am cosidering only the best users (atleast 200 books feedback) group
them based on the book title.

y = filtered_ratings.groupby('Book-Title').count()

# The above dataframe tell how many users have read the book.

y.sort_values('User-ID',ascending=False)

y = y['User-ID'] >= 50

famous_books = y[y].index

final_ratings = filtered_ratings[filtered_ratings['Book-
Title'].isin(famous_books)]

# pivot table giving the ratings for each book from each user

# Book row with userid as column

pt = final_ratings.pivot_table(index='Book-Title', columns='User-
ID',values='Book-Rating')

pt = pt.fillna(0)

similarity_scores = cosine_similarity(pt)

df_temp = pd.DataFrame(similarity_scores)

def recommend(book_name):

    index = np.where(pt.index == book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])),
key=lambda x : x[1], reverse=True)[1:6]

    # Lets create empty list and in that lies i want ot populate with the book
information

    # Book author book-title image url

    # Empty list

    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

recommend('Animal Farm')
```

```
# Import Pickle and dump the data and models

import pickle as pkl

pkl.dump(popular_df,open('popular.pkl','wb')) # Popularity based
recommender system

pkl.dump(books,open('books.pkl','wb')) # book data

pkl.dump(pt,open('pt.pkl','wb')) # books and user feedback

pkl.dump(similarity_scores, open('similarity_scores.pkl','wb'))
```

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Embedded Streamlit App</title>

</head>
```

```
<body>

   <!-- <h1>My Streamlit App in HTML</h1> -->

    <div style="display: flex;">

      <iframe src="http://localhost:8501" style="border:none; height: 100vh;
margin: -8px; width: 75vw;"></iframe>



         <iframe

         src="https://www.chatbase.co/chatbot-
iframe/zLRoWQvCL8Zj8plFgVXeK"

         style="height: 100vh; width: 25vw; margin: -8px;"

         frameborder="0"



      ></iframe>

    </div>



</body>

</html>
```