Final Project

CPRA

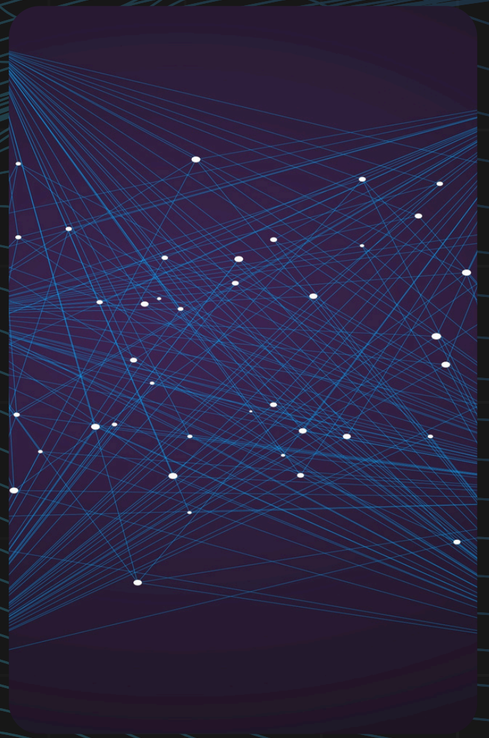
M01 Book Recommendation System

Presented by
Min Aung Thu Hein Htut

Introduction

M01 is a book recommendation engine that utilizes collaborative-filtering system to provide suitable recommendations based on the chosen book of a user.

The current GUI of this recommendation engine is web-based. However, the system itself can be integrated into other applications as well.



Problem Statement

Nowadays, due to the rapid rise of information and data, people are sometimes overwhelmed by the sheer amount of choices in the online retail sectors like e-commerce, movie streaming websites, etc. It is difficult for people to make a proper decision due to the large amount of choices. The same situation happens for book readers. The process to find books that are relevant to their taste can be time-consuming and annoything at times.

Solution

M01 can provide relevant and helpful book recommendations based on the user's choice. M01 leverages collaborative filtering techniques to match the user's preference with other similar users and provide recommendations based on that. This system can be used in digital libraries, online e-commerce platforms.

Target Audience

The direct target audience is book readers who want to purchase books online or rent books online. The indirect target audience will be online book stores, digital libraries and universities.

Key Features

- Intelligent Book Recommendations - Uses collaborative filtering to suggest books similar to the one selected by the user.
- Real-time Suggestions - Provides instant recommendations using a trained machine learning model (K-Nearest Neighbors).
- Simple Book Selection Interface - Allows users to choose a book from a dropdown list — no typing required.
- Card-based Recommendation Display - Shows suggestions as elegant cards featuring book cover, title, and author.
- Image Integration via URLs - Dynamically loads high-resolution book cover images using image URLs from the dataset.

Code Explanation

Part 1 - Collaborative-filtering system

```
[2]: import pandas as pd
     import numpy as np
```

```
[3]: book_path = "/Users/minaungthu/Documents/KU /1st year/2nd Semester/computer programming/project/dataset/Books.csv"
     user_path = "/Users/minaungthu/Documents/KU /1st year/2nd Semester/computer programming/project/dataset/Users.csv"
     rating_path = "/Users/minaungthu/Documents/KU /1st year/2nd Semester/computer programming/project/dataset/Ratings.csv"

     book_df = pd.read_csv(book_path)
     user_df = pd.read_csv(user_path)
     rating_df = pd.read_csv(rating_path)
```

```
/var/folders/zb/bss23kdn25bg5_5cnl9dyv600000gn/T/ipykernel_10106/586059617.py:5: DtypeWarning: Columns (3) have mixed types. Specify dtype op
tion on import or set low_memory=False.
  book_df = pd.read_csv(book_path)
```

```
[4]: book_df.columns
```

```
[4]: Index(['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher',
       'Image-URL-S', 'Image-URL-M', 'Image-URL-L'],
      dtype='object')
```

```
[5]: book_df = book_df[['ISBN','Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher','Image-URL-L']]
```

```
[6]: book_df.columns
```

```
[6]: Index(['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher',
       'Image-URL-L'],
      dtype='object')
```

```
[7]: user_df.columns
     user_df.rename(columns={"User-ID":'user_id',"Location":"location", "Age":'age'},inplace=True)
     user_df.columns
```

```
[7]: Index(['user_id', 'location', 'age'], dtype='object')
```

```
[8]: rating_df.columns
```

```
[8]: Index(['User-ID', 'ISBN', 'Book-Rating'], dtype='object')
```

- Import libraries (module concept)
- Load 3 datasets with the pandas module
- Drop unnecessary columns and rename some columns for easier data handling for book_df, user_df.

[8]:
```
rating_df.columns
```

[8]:
```
Index(['User-ID', 'ISBN', 'Book-Rating'], dtype='object')
```

[9]:
```
rating_df.rename(columns={"User-ID":"user_id", "Book-Rating":"rating"},inplace=True)
rating_df.columns
```

[9]:
```
Index(['user_id', 'ISBN', 'rating'], dtype='object')
```

[10]:
```
print(f"book_df size: {book_df.shape} \nuser_df size : {user_df.shape} \nrating_df size: {rating_df.shape}")
```

```
book_df size: (271360, 6)
user_df size : (278858, 3)
rating_df size: (1149780, 3)
```

[11]:
```
#storing users who rated at least 100 books
x = rating_df['user_id'].value_counts() > 100
```

[12]:
```
x[x].shape
```

[12]:
```
(1825,)
```

[13]:
```
y= x[x].index
y
```

[13]:
```
Index([ 11676, 198711, 153662,  98391,  35859, 212898, 278418,  76352, 110973,
       235105,
       ...
       238186,  99441, 187410, 262070,  70183,  40553,  39345, 266283, 189666,
       140879],
      dtype='int64', name='user_id', length=1825)
```

[14]:
```
rating_df = rating_df[rating_df['user_id'].isin(y)]
rating_df.head()
```

[14]:

	user_id	ISBN	rating
412	276925	0006511929	0
413	276925	002542730X	10

- Rename the columns in rating_df dataset.
- Check the dimensions of each dataframe.
- Filter users who have at least recommended 100 books, for the significance of data
- Filter rating_df based on the books which the selected users have recommended

```
[15]: rating_df.shape
```

```
[15]: (656605, 3)
```

```
[16]: books_with_ratings = rating_df.merge(book_df, on='ISBN')
books_with_ratings.head()
```

[16]:

	user_id	ISBN	rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-L
0	276925	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	http://images.amazon.com/images/P/002542730X.0...
1	276925	0060520507	0	Sushi for Beginners : A Novel (Keyes, Marian)	Marian Keyes	2003	William Morrow	http://images.amazon.com/images/P/0060520507.0...
2	276925	0060930934	0	Wasted : A Memoir of Anorexia and Bulimia	Marya Hornbacher	1999	Perennial	http://images.amazon.com/images/P/0060930934.0...
3	276925	0060951303	0	La casa de los espÃritus	Isabel Allende	1995	Rayo	http://images.amazon.com/images/P/0060951303.0...
4	276925	0140154078	6	The Music of Chance	Paul Auster	1993	Penguin Books	http://images.amazon.com/images/P/0140154078.0...

```
[17]: number_rating = books_with_ratings.groupby('Book-Title')['rating'].count().reset_index()
```

```
[18]: number_rating.head()
```

[18]:

	Book-Title	rating
0	A Light in the Storm: The Civil War Diary of ...	3
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Beyond IBM: Leadership Marketing and Finance ...	1
4	Clifford Visita El Hospital (Clifford El Gran...	1

- Merge new rating_df and book_df to get a new dataframe for books with ratings.
- Create a new dataframe “number_rating” by grouping each book with the total number of ratings.

```
[19]: number_rating.rename(columns={'rating':'num_of_rating'},inplace=True)

[20]: number_rating.head()
```

[20]:

	Book-Title	num_of_rating
0	A Light in the Storm: The Civil War Diary of ...	3
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Beyond IBM: Leadership Marketing and Finance ...	1
4	Clifford Visita El Hospital (Clifford El Gran...	1

```
[21]: final_rating = books_with_ratings.merge(number_rating, on='Book-Title')
final_rating.head()
```

[21]:

	user_id	ISBN	rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-L	num_of_rating
0	276925	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	http://images.amazon.com/images/P/002542730X.0...	105
1	276925	0060520507	0	Sushi for Beginners : A Novel (Keyes, Marian)	Marian Keyes	2003	William Morrow	http://images.amazon.com/images/P/0060520507.0...	16
2	276925	0060930934	0	Wasted : A Memoir of Anorexia and Bulimia	Marya Hornbacher	1999	Perennial	http://images.amazon.com/images/P/0060930934.0...	12
3	276925	0060951303	0	La casa de los espÃritus	Isabel Allende	1995	Rayo	http://images.amazon.com/images/P/0060951303.0...	8
4	276925	0140154078	6	The Music of Chance	Paul Auster	1993	Penguin Books	http://images.amazon.com/images/P/0140154078.0...	6

```
[22]: #taking books with at least 30 ratings
```

- Create a new dataframe called final_rating by adding the “number_rating” dataframe into the “books_with_ratings” dataframe.

[22]:
```
#taking books with at least 30 ratings
final_rating = final_rating[final_rating['num_of_rating'] >= 30]
final_rating.head()
```

[22]:

	user_id	ISBN	rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-L	num_of_rating
0	276925	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	http://images.amazon.com/images/P/002542730X.0...	105
5	276925	0140327592	0	Matilda	Roald Dahl	1990	Viking Penguin Inc	http://images.amazon.com/images/P/0140327592.0...	79
12	276925	0316666343	0	The Lovely Bones: A Novel	Alice Sebold	2002	Little, Brown	http://images.amazon.com/images/P/0316666343.0...	430
15	276925	0385504209	8	The Da Vinci Code	Dan Brown	2003	Doubleday	http://images.amazon.com/images/P/0385504209.0...	335
32	276925	0679745580	0	In Cold Blood (Vintage International)	TRUMAN CAPOTE	1994	Vintage	http://images.amazon.com/images/P/0679745580.0...	30

[23]:
```
final_rating.shape
```

[23]:
```
(148590, 9)
```

[24]:
```
final_rating.drop_duplicates(['user_id','Book-Title'],inplace=True)
final_rating.shape
```

[24]:
```
(145469, 9)
```

[25]:
```
book_pivot = final_rating.pivot_table(columns='user_id', index='Book-Title', values= 'rating')
book_pivot
```

[25]:

user_id	183	254	507	882	1424	1435	1733	1903	2033	2110	...	276018	276463	276680	276925	277427	277478	277639	278137	278188
Book-Title																				
10 Lb. Penalty	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Filter the final_rating dataframe by only choosing books with at least 30 ratings.
- Remove the duplicates in final_rating dataframe.
- Create a table called book_pivot which will be used to train the model.

```
[25]: book_pivot = final_rating.pivot_table(columns='user_id', index='Book-Title', values= 'rating')
      book_pivot
```

[25]:

user_id	183	254	507	882	1424	1435	1733	1903	2033	2110	...	276018	276463	276680	276925	277427	277478	277639	278137	278188
Book-Title																				
10 Lb. Penalty	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16 Lighthouse Road	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
1984	NaN	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1st to Die: A Novel	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2010: Odyssey Two	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
You Belong to Me and Other True Cases (Ann Rule's Crime Files: Vol. 2)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Zlata's Diary: A Child's Life in Sarajevo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Zoya	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
\O\" Is for Outlaw"	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- A closer look at the book_pivot which has a lot of NaN values.

```
[26]: book_pivot.shape

[26]: (2372, 1804)

[40]: book_pivot.fillna(0, inplace=True)
      book_pivot
```

[40]:

user_id	183	254	507	882	1424	1435	1733	1903	2033	2110	...	276018	276463	276680	276925	277427	277478	277639	278137	278188
Book-Title																				
10 Lb. Penalty	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
16 Lighthouse Road	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
1984	0.0	9.0	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
1st to Die: A Novel	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
2010: Odyssey Two	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
You Belong to Me and Other True Cases (Ann Rule's Crime Files: Vol. 2)	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
Zlata's Diary: A Child's Life in Sarajevo	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
Zova	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

- Replace null values with 0.

[28]:
```
!pip install scipy
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: scipy in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (1.15.2)
Requirement already satisfied: numpy<2.5,>=1.23.5 in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (from scipy)
(2.2.3)
```

[29]:
```
#since the dataset is mostly sparse, we will apply sparse matrix
from scipy.sparse import csr_matrix
book_sparse = csr_matrix(book_pivot)
type(book_sparse)
```

[29]:
```
scipy.sparse._csr.csr_matrix
```

[30]:
```
!pip install scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: scikit-learn in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (from scikit-learn)
(2.2.3)
Requirement already satisfied: scipy>=1.6.0 in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (from scikit-learn)
(1.15.2)
Requirement already satisfied: joblib>=1.2.0 in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (from scikit-learn)
(1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /Users/minaungthu/miniforge3/envs/opencv-env/lib/python3.13/site-packages (from scikit
-learn) (3.6.0)
```

[31]:
```
from sklearn.neighbors import NearestNeighbors
#using brute force since the input data is very sparse
model = NearestNeighbors(algorithm= 'brute')
```

[32]:
```
model.fit(book_sparse)
```

[32]:
```
NearestNeighbors
NearestNeighbors(algorithm='brute')
```

[33]:
```
distance, suggestion = model.kneighbors(book_pivot.iloc[1,:].values.reshape(1,-1), n_neighbors=6 )
```

- Use scipy and create a sparse matrix out of book_pivot dataframe.
- Utilize NeastNeigbors classifier with brute algorithm to train the model.

[33]:
```
distance, suggestion = model.kneighbors(book_pivot.iloc[1,:].values.reshape(1,-1), n_neighbors=6 )
```

[34]:
```
for i in suggestion[0]:
    print(book_pivot.index[i])
```

```
16 Lighthouse Road
Hurricane Bay
Acts of Love
About Face
Everything to Gain
Stay Out of the Basement (Goosebumps, No 2)
```

[35]:
```
def recommend_book(book_name):
    book_id = np.where(book_pivot.index == book_name)[0][0]
    distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1), n_neighbors=6 )

    for i in range(len(suggestion)):
            books = book_pivot.index[suggestion[i]]
            for j in books:
                if j == book_name:
                    print(f"You searched '{book_name}'\n")
                    print("The suggested books are: \n")
                else:
                    print(j)
```

[36]:
```
book_pivot.head(10)
```

[36]:

user_id	183	254	507	882	1424	1435	1733	1903	2033	2110	...	276018	276463	276680	276925	277427	277478	277639	278137	278188	27
Book-Title																					
10 Lb. Penalty	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	
16 Lighthouse Road	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	

- Test the suggestion from the model by using for loop.
- Create recommend_book function to search by book names.
- book_id is an integer index number which will be used to search the row.
- model.kneighbors takes 2-D arrays as input, but book_pivot.iloc[book_id,:] only returns 1-D array. So it has to be reshaped.

Lane																			
2061: Odyssey Three	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
24 Hours	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	10.0	0.0	0.0	0.0	0.0
2nd Chance	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
311 Pelican Court	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

10 rows × 1804 columns

[37]:
```
recommend_book("1984")
```

```
You searched '1984'

The suggested books are:

Lying Awake
The Fourth Deadly Sin
Acts of Love
Another City, Not My Own: A Novel in the Form of a Memoir
Everything to Gain
```

[38]:
```
book_names = book_pivot.index
```

[39]:
```
import pickle
pickle.dump(model,open('specs/model.pkl','wb'))
pickle.dump(book_names,open('specs/book_names.pkl','wb'))
pickle.dump(final_rating,open('specs/final_rating.pkl','wb'))
pickle.dump(book_pivot,open('specs/book_pivot.pkl','wb'))
```

- Now, the model is ready to be utilized, so we will use pick module to serialize the data and store model, book_names, and two dataframes.

Code Explanation

Part 2 - Light-weight Website

Imports and Overview

- Imports required libraries:

```
import streamlit as st       # For web interface
import pickle                # To load saved model/data
import numpy as np           # For numeric operations
import pandas as pd          # For data handling
```

- Purpose:

- These libraries enable the core app functionality: UI, ML model, and data.

Book Recommender Class - Initialization

- A class that wraps recommendation logic.

```
class BookRecommender:
    def __init__(self, model, book_pivot, final_rating):
        self.model = model
        self.book_pivot = book_pivot
        self.final_rating = final_rating
```

- Why?

- Object-oriented design makes the code organized and reusable.

Inside the class - recommend_books Method

- Finds similar books using the ML model.

```python
def recommend_books(self, book_name):
    if book_name not in self.book_pivot.index:
        return []
    book_id = np.where(self.book_pivot.index == book_name)[0][0]
    distance , suggestions = self.model.kneighbors(self.book_pivot.iloc[book_id, :].values.reshape(1, -1), n_neighbors=6)

    recommended_books = []
    for i in range(len(suggestions)):
        books = self.book_pivot.index[suggestions[i]]
        for j in books:
            if j != book_name:
                recommended_books.append(j)
    return recommended_books
```

- Uses: Selection (if), Loop (for)

Inside the class - get_book_details Method

- Retrieves author and image URL for a book.

```python
def get_book_details(self, title):
    row = self.final_rating[self.final_rating['Book-Title'] == title]
    if not row.empty:
        return {
            "author": row['Book-Author'].values[0],
            "image_url": row['Image-URL-L'].values[0]
        }
    return {"author": "Unknown", "image_url": None}
```

Load Pickled Model and Data

- Load saved model and datasets:

```
model = pickle.load(open('specs/model.pkl', 'rb'))
book_names = pickle.load(open('specs/book_names.pkl', 'rb'))
final_rating = pickle.load(open('specs/final_rating.pkl', 'rb'))
book_pivot = pickle.load(open('specs/book_pivot.pkl', 'rb'))
recommender = BookRecommender(model, book_pivot, final_rating)
```

Streamlit UI Setup, User Input & Button

- Configure Streamlit page, title, buttons and provide recommendation

```python
#UI
st.set_page_config(page_title="Books Recommender", layout="wide", page_icon="📚")
st.title(" M01 Book Recommendation System")
st.write("Select a book to get ML-powered suggestions!")
selected_book = st.selectbox("Choose a book you have read before:", book_names)

if selected_book and st.button("Show Recommendations"):
    with st.spinner("Finding similar books..."):
        recommended_books = recommender.recommend_books(selected_book)
        time.sleep(2)

    if recommended_books:
        st.success(f"Books similar to: *{selected_book}*")
        cols = st.columns(5)

        for idx, book in enumerate(recommended_books):
            details = recommender.get_book_details(book)
            with cols[idx % 5]:
                st.image(details["image_url"], use_container_width=True)
                st.markdown(f"**{book}**")
                st.caption(f"By: {details['author']}")
    else:
        st.warning("No recommendations found.")
```

Technology Stack

Layer	Technology / Tool	Purpose
Frontend	Streamlit	Interactive web interface for users to select books
Backend	Python	Core logic and orchestration of recommendation process
	Object-Oriented Programming	Modular and maintainable design using classes and functions
Machine Learning	Scikit-learn (KNN)	Collaborative filtering using nearest neighbors
	Brute-force Search	Required for sparse, high-dimensional user-book matrix
Data Handling	Pandas	Reading, merging, and cleaning CSV data
	NumPy	Numerical computation, indexing, reshaping matrices
	CSV Files	Book, user, and rating data stored and loaded from CSVs
Data Engineering	Pivot Tables (Pandas)	User-book rating matrix creation
	Sparse Matrices (scipy)	Memory-efficient representation of ratings
Model Storage	Pickle	Saving and loading pre-trained models and processed data
Virtual Environment	Conda Virtual Environment	Control versions and dependencies

MOJ