Terna Engineering College

Computer Engineering Department

Class: BE **Sem.: VII**

Course: Natural Language Processing [NLP]

Experiment No. 09- MINI PROJECT

PART A

(PART A : TO BE REFFERED BY STUDENTS)

A.1 Aim: Book Recommendation System

A.2 Prerequisite: Python, Flask, scikit-learn, Pandas, Numpy, Google Colab, Pickle libraries.

A.3 Outcome:

Students will learn how to implement a book recommendation system using popularity-based recommendations and collaborative filtering. The project demonstrates the application of NLP and machine learning techniques to personalize user experiences in web applications.

A.4 Theory:

The Book Recommendation System is designed to assist users in discovering new books by providing personalized recommendations based on their past behavior or general popularity. Recommendation systems are integral to modern web applications, and this project focuses on two primary recommendation strategies: popularity-based and collaborative filtering. Using Natural Language Processing (NLP) and machine learning, the system analyzes user interactions to generate suggestions. The backend is built using Flask, while Python handles the machine learning algorithms.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

Roll. No.: 28	**Name:** Harsh A. Minde.
Class: BE COMP C	**Batch:** C2
Date of Experiment:	**Date of Submission:**
Grade:	

B.1 Software Code written by student:

The **Book Recommendation System** integrates Python, Flask, and machine learning models to create a dynamic and interactive platform for suggesting books based on popularity and collaborative filtering. This project demonstrates the successful implementation of two recommendation strategies and showcases the combined power of web development and data science.

Project Overview:

The code is designed to offer users recommendations based on two main approaches: popularity-based suggestions and collaborative filtering. Flask is used as the backend framework to handle user interactions, while pre-trained machine learning models handle the recommendation logic. The models and data, preprocessed and serialized using pickle, ensure the system operates efficiently and responds quickly to user queries.

Explanation of Code Functionality:

1. **Initialization and Libraries:**

   The project begins by importing necessary libraries, including Flask, pickle, numpy, and machine learning utilities. The serialized models (stored using pickle) are loaded at runtime to provide the recommendation logic. This ensures that the system avoids retraining models on every execution, leading to faster and more efficient responses.

2. **Popularity-Based Recommendations (Homepage):**

   The homepage uses a popularity-based approach to list the top 50 books based on user feedback (number of ratings and average rating). This is a static recommendation that does not change with individual users but offers a snapshot of popular content across the entire dataset. This information is served using Flask's / route, which loads data from the pre-trained popularity model (popular.pkl) and displays it using HTML templates (index.html).

   - **Purpose:** To allow users to see top books that have received widespread attention, providing an easy entry point for book discovery.
   - **Frontend Integration:** The index.html template renders this data in a responsive layout using Bootstrap, ensuring it adapts well to different screen sizes.

3. **Collaborative Filtering-Based Recommendations:**

   The real value of this system is derived from its personalized recommendations, which are generated using collaborative filtering. Collaborative filtering works by analyzing similarities between users and books based on past interactions (ratings). When a user inputs a book title on the recommendation page (/recommend_books route), the system uses cosine similarity to find books that are most similar to the selected title. The similarity scores are precomputed and stored in a file (similarity_scores.pkl) for efficient retrieval.

   - **Purpose:** To recommend books that are likely to align with a user's taste, based on their previous choices or similar users' preferences.
   - **Backend Logic:** Flask handles the user's input, processes it using the precomputed similarity matrix, and returns a list of similar books. This dynamic output is presented to the user on the recommend.html page.

4. **Data Preprocessing and Models:**

   Preprocessing and training of the models occur offline, allowing the web app to serve pre-built models for recommendation generation. The data (ratings, books, and user preferences) are cleaned, normalized, and processed into matrices that allow the system to compute similarities efficiently. The pivot table (pt.pkl) and similarity matrix ensure that recommendations are tailored based on user interaction with the books.

5. **Frontend Display:**

   The project leverages Bootstrap to create a clean and interactive frontend where book recommendations are displayed. Users can input their book preferences and receive suggestions in real-time, improving user engagement. The responsive design ensures that the layout adapts well across devices, making the system accessible to a wide audience.

6. **Program Flow:**
   - The user navigates to the homepage, where they see a list of the top 50 books based on popularity.
   - On the recommendation page, they enter the name of a book they like. The system processes the input, calculates similar books based on collaborative filtering, and displays them in a user-friendly format.
   - The system provides an engaging experience by dynamically generating personalized recommendations, enhancing book discovery for the user.

App.py :

app.py

book-recommender-system > app.py > recommend

```python
from flask import Flask,render_template,request
import pickle
import numpy as np

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'))

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html',
                           book_name = list(popular_df['Book-Title'].values),
                           author=list(popular_df['Book-Author'].values),
                           image=list(popular_df['Image-URL-M'].values),
                           votes=list(popular_df['num_ratings'].values),
                           rating=list(popular_df['avg_rating'].values)
                           )

@app.route('/recommend')
def recommend_ui():
    return render_template('recommend.html')

@app.route('/recommend_books',methods=['post'])
def recommend():
    user_input = request.form.get('user_input')

    # Check if user_input is empty
    if not user_input:
        return render_template('recommend.html', data=[], error_message="No results found or please enter a valid book name.")

    # Try to find the index of the book title
    try:
        index = np.where(pt.index == user_input)[0][0]
    except IndexError:
```

app.py

book-recommender-system > app.py > recommend

```python
def recommend():
    try:
        index = np.where(pt.index == user_input)[0][0]
    except IndexError:
        # If the book title is not found, display an error message
        return render_template('recommend.html', data=[], error_message="No results found or please enter a valid book name.")

    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)

    print(data)

    return render_template('recommend.html',data=data)

if __name__ == '__main__':
    app.run(debug=True)
```

Index.html :

index.html

book-recommender-system > templates > index.html > html > head

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book Recommender System</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
          integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
    <!-- Custom CSS -->
    <style>
        body {
            background-color: #f4f4f9;
            font-family: 'Arial', sans-serif;
            margin-top: 60px; /* Adjusted margin */
        }


        .navbar {
            background-color: #007bff;
            position: fixed; /* Makes the navbar fixed */
            top: 0; /* Aligns it to the top */
            left: 0; /* Aligns it to the left */
            right: 0; /* Ensures it covers the full width */
            z-index: 1000; /* Ensures it stays above other content */
        }

        .navbar-brand, .navbar-nav li a {
            color: white !important;
            font-size: 1.8rem;
            font-weight: bold;
        }

        h1 {
            color: #343a40;
            font-weight: 700;
```

index.html

book-recommender-system > templates > index.html > html > head

```
<html lang="en">
<head>
    <style>
        h1 {
            color: #343a40;
            font-weight: 700;
            text-align: center;
            margin-top: 40px;
            font-size: 2.5rem;
        }

        .book-card {
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            padding: 20px;
            margin-bottom: 30px;
            text-align: center;
            height: 350px; /* Fixed height for uniformity */
            display: flex;
            flex-direction: column;
            justify-content: space-between;
        }

        .book-card img {
            max-width: 100%;
            height: 200px; /* Fixed image height */
            object-fit: cover; /* Ensure image fills the space */
            border-radius: 5px;
        }

        .book-card h4, .book-card p {
            color: #007bff;
            font-weight: bold;
        }
```

index.html

book-recommender-system > templates > index.html > html > body > nav.navbar.navbar-expand-lg > ul.nav.navbar-nav

```
<html lang="en">
<head>
    <style>
        }

        .book-card h4 {
            font-size: 1.2rem;
        }

        .book-card p {
            font-size: 1rem;
        }

        .book-card .votes, .book-card .rating {
            color: #6c757d;
            font-size: 0.9rem;
        }

        .row {
            display: flex;
            flex-wrap: wrap;
            gap: 20px; /* Ensure consistent spacing between cards */
        }

        .col-md-3 {
            flex: 1 1 calc(25% - 20px); /* Ensures 4 columns with gaps */
            box-sizing: border-box;
        }
    </style>
</head>
<body>

    <nav class="navbar navbar-expand-lg">
        <a class="navbar-brand" href="#">Book Recommender</a>
        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
```

index.html

book-recommender-system > templates > index.html > html > body > nav.navbar.navbar-expand-lg > ul.nav.navbar-nav

```
<html lang="en">
<body>
    <nav class="navbar navbar-expand-lg">
        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/recommend">Recommend</a></li>
            <!-- <li><a href="#">Contact</a></li> -->
        </ul>
    </nav>


    <!-- <nav class="navbar navbar-expand-lg">
        <div class="container">
            <a class="navbar-brand" href="#">Book Recommender</a>
            <div class="collapse navbar-collapse">
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="/">Home</a></li>
                    <li><a href="/recommend">Recommend</a></li>
                </ul>
            </div>
        </div>
    </nav> -->


    <div class="container">
        <h1>Top 50 Books</h1>
        <div class="row">
            {% for i in range(book_name|length) %}
            <div class="col-md-3">
                <div class="book-card">
                    <img src="{{ image[i] }}" alt="Book Image">
                    <h4>{{ book_name[i] }}</h4>
                    <p>{{ author[i] }}</p>
                    <p class="votes">Votes: {{ votes[i] }}</p>
                    <p class="rating">Rating: {{ rating[i] }}</p>
                </div>
```

index.html

book-recommender-system > templates > index.html > html > body > div.container > div.row > div.col-md-3

```
<html lang="en">
<body>
    <div class="container">
        <div class="row">
            <div class="col-md-3">
                    <p>{{ author[i] }}</p>
                    <p class="votes">Votes: {{ votes[i] }}</p>
                    <p class="rating">Rating: {{ rating[i] }}</p>
                </div>
            </div>
            {% endfor %}
        </div>
    </div>

</body>
</html>
```

Recommend.html :

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style > .book-card

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book Recommender System</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
          href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
          integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
          crossorigin="anonymous">
    <!-- Custom CSS -->
    <style>
        body {
            background-color: #f4f4f9;
            font-family: 'Arial', sans-serif;
            margin-top: 60px; /* Adjusted margin */
        }

        .navbar {
            background-color: #007bff;
            position: fixed; /* Makes the navbar fixed */
            top: 0; /* Aligns it to the top */
            left: 0; /* Aligns it to the left */
            right: 0; /* Ensures it covers the full width */
            z-index: 1000; /* Keeps it on top */
        }

        .navbar-brand, .navbar-nav li a {
            color: white !important;
            font-size: 1.8rem;
            font-weight: bold;
        }

        h1 {
            color: #343a40;
            font-weight: 700;
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style

```
<html lang="en">
<head>
    <style>
        h1 {
            color: #343a40;
            font-weight: 700;
            text-align: center;
            margin-top: 40px;
            font-size: 2.5rem;
        }

        .book-card {
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            padding: 20px;
            margin-bottom: 30px;
            text-align: center;
            height: 350px; /* Fixed height for uniformity */
            display: flex;
            flex-direction: column;
            justify-content: space-between;
        }

        .book-card img {
            max-width: 100%;
            height: 200px; /* Fixed image height */
            object-fit: cover;
            border-radius: 5px;
        }

        .book-card h4, .book-card p {
            color: #007bff;
            font-weight: bold;
        }

        .row {
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style

```
<html lang="en">
<head>
    <style>
        }

        .row {
            display: flex;
            flex-wrap: wrap;
            justify-content: space-between; /* Ensures cards take up space proportionally */
        }

        .col-md-3 {
            flex: 1 1 calc(25% - 20px);
            box-sizing: border-box;
            margin-bottom: 20px;
        }

        /* Custom styles for form */
        .form-container {
            margin-top: 40px;
            display: flex;
            justify-content: center;
            align-items: center; /* Center items vertically */
        }

        .search-bar {
            display: flex;
            align-items: center;
            position: relative;
            flex: 1; /* Makes the search bar expand to take available space */
        }

        .search-bar input {
            padding-left: 40px;
            height: 50px; /* Adjusted height */
            border-radius: 25px; /* Rounded corners */
            border: 2px solid #007bff; /* Border color */
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style > .book-card h4 > .book-card p

```
<html lang="en">
<head>
    <style>
        .search-bar input {
            border-radius: 25px; /* Rounded corners */
            border: 2px solid #007bff; /* Border color */
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /* Shadow effect */
        }

        .search-bar .search-icon {
            position: absolute;
            left: 10px;
            top: 50%; /* Center vertically */
            transform: translateY(-50%); /* Adjust position */
            font-size: 18px;
            color: #007bff;
        }

        .btn-clear, .btn-recommend {
            border-radius: 25px; /* Rounded corners */
            height: 50px; /* Adjusted height */
            padding: 0 20px; /* Added horizontal padding for better size */
            margin-left: 10px; /* Space between buttons */
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2); /* Shadow effect */
        }

        .btn-clear {
            background-color: #dc3545;
            color: white;
        }

        .btn-recommend {
            background-color: #007bff;
            color: white;
        }

        /* Error and popup messages */
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style > .book-card h4 > .book-card p

```
<html lang="en">
<head>
    <style>
        .alert {
            margin-top: 20px;
            display: none; /* Hidden by default */
            text-align: center; /* Center the text */
            width: 80%; /* Adjust width as needed */
            margin-left: auto; /* Center horizontally */
            margin-right: auto; /* Center horizontally */
        }
    </style>
</head>
<body>

    <nav class="navbar navbar-expand-lg">
        <a class="navbar-brand" href="#">Book Recommender</a>
        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/recommend">Recommend</a></li>
            <!-- <li><a href="#">Contact</a></li> -->
        </ul>
    </nav>

    <div class="container">
        <h1>Recommend Books</h1>
        <div class="form-container">
            <form id="searchForm" action="/recommend_books" method="post" class="form-inline"
                style="display: flex; align-items: center;">
                <div class="search-bar">
                    <span class="glyphicon glyphicon-search search-icon"></span>
                    <input id="bookInput" name="user_input" type="text" class="form-control"
                        placeholder="Enter a book name" value="{{ user_input }}">
                </div>
                <input type="submit" class="btn btn-recommend btn-lg" value="Get Recommendations">
                <button type="button" id="clearBtn" class="btn btn-clear btn-lg">Clear</button>
            </form>
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style > .book-card h4 > .book-card p

```
<html lang="en">
<body>
    <div class="container">
        <div class="form-container">
            </form>
        </div>

        <!-- Error message for invalid or empty input -->
        <!-- <div class="alert alert-danger" id="errorMessage">
            <strong>NOTE:</strong> No results found or please enter a valid book name.
        </div> -->

        <!-- Error message for invalid or empty input -->
        {% if error_message %}
        <div class="alert alert-danger" id="errorMessage">
            <strong>NOTE:</strong> {{ error_message }}
        </div>
        {% endif %}

        <div class="row">
            {% if data %}
            {% for i in data %}
            <div class="col-md-3">
                <div class="book-card">
                    <img src="{{i[2]}}" alt="Book Image">
                    <h4>{{i[0]}}</h4>
                    <p>{{i[1]}}</p>
                </div>
            </div>
            {% endfor %}
            {% endif %}
        </div>
    </div>

    <!-- Script for validation and handling errors -->
    <script>
        document.getElementById('searchForm').addEventListener('submit', function(event) {
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style > .book-card h4 > .book-card p

```
  2   <html lang="en">
141   <body>
195       <script>
196           document.getElementById('searchForm').addEventListener('submit', function(event) {
197               const userInput = document.getElementById('bookInput').value.trim();
198               if (!userInput) {
199                   event.preventDefault();
200                   document.getElementById('errorMessage').style.display = 'block';
201               }
202           });
203
204           // document.getElementById('clearBtn').addEventListener('click', function() {
205           //     // Reload the page to clear all fields and results
206           //     // location.reload();
207
208           //     document.getElementById('bookInput').value = '';          // Clear the input
209           //     document.getElementById('errorMessage').style.display = 'none';       // Hide error message
210
211           //     // Clear the displayed results
212           //     const rowDiv = document.querySelector('.row');
213           //     rowDiv.innerHTML = ''; // Clear all contents of the row
214
215           //     while (rowDiv.firstChild) {
216           //         rowDiv.removeChild(rowDiv.firstChild); // Remove all child elements (book cards)
217           //     }
218
219           //     // Optionally, reset the form to its initial state
220           //     // document.getElementById('searchForm').reset(); // This resets the form fields, if any
221
222           //     // Delay the reload for a brief moment to ensure everything is cleared
223           //     setTimeout(function() {
224           //         location.reload();  // Reload the page to clear all fields and results
225           //     }, 1); // Adjust the timeout duration if necessary
226
227           // });
228
229           document.getElementById('clearBtn').addEventListener('click', function() {
```

recommend.html 7

book-recommender-system > templates > recommend.html > html > head > style > .book-card h4 > .book-card p

```
  2   <html lang="en">
141   <body>
195       <script>
228
229           document.getElementById('clearBtn').addEventListener('click', function() {
230               // Clear the input field
231               document.getElementById('bookInput').value = '';
232
233               // Hide the error message if it exists
234               const errorMessage = document.getElementById('errorMessage');
235               if (errorMessage) {
236                   errorMessage.style.display = 'none'; // Hide error message
237               }
238
239               // Clear the displayed results
240               const rowDiv = document.querySelector('.row');
241               if (rowDiv) {
242                   rowDiv.innerHTML = ''; // Clear all contents of the row
243               }
244           });
245
246
247           // If no data is found, display the 'noResultsMessage'
248           {% if not data %}
249           document.getElementById('noResultsMessage').style.display = 'block';
250           {% endif %}
251       </script>
252
253   </body>
254   </html>
255
```

B.2 Input and Output:

Input Process:

1. **Homepage Interaction:**
   - Upon landing on the homepage (index.html), the user is presented with a static list of the top 50 books. This list is derived from the popularity-based model, which calculates the books with the highest number of ratings and best average ratings.
   - **Example Input:**

     A user views the homepage and browses through the top 50 books. Each book's details, including title, author, rating, and votes, are displayed in an organized grid.

2. **Recommendation Input:**
   - On the recommendation page (recommend.html), users are prompted to enter the name of a book. For instance, the user might type "Harry Potter" into the search bar. This input is passed to the Flask backend, which queries the collaborative filtering model for similar books.
   - **Example Input:**

     The user types in a book title like "Harry Potter" into the search bar and submits the form.

Output Process:

1. **Popularity-Based Output:**
   - On the homepage, the user sees a pre-generated list of the top 50 books. These are presented with details such as book title, author, rating, and number of votes. This output is static and consistent for all users, as it reflects general trends rather than personalized suggestions.

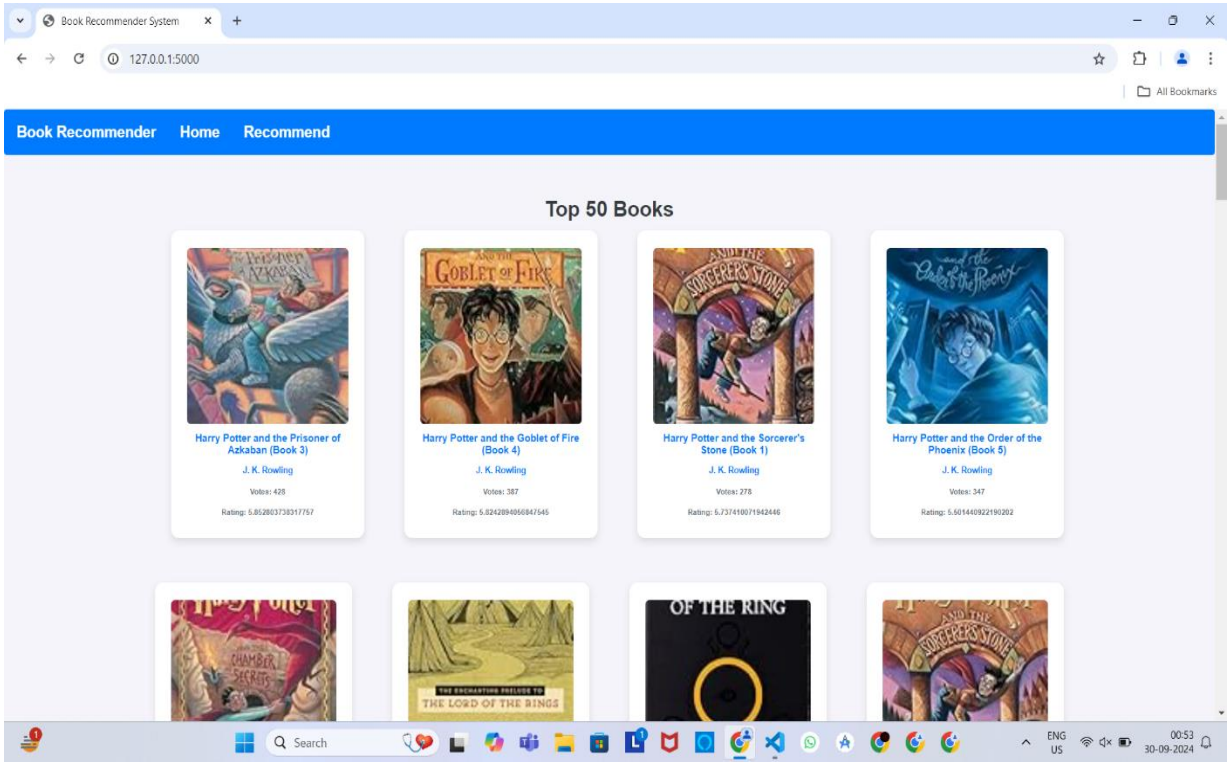
- **Example Output:**

  The user sees "The Alchemist" by Paulo Coelho with a rating of 4.7 and 1200 votes. This provides an overview of popular choices within the system.

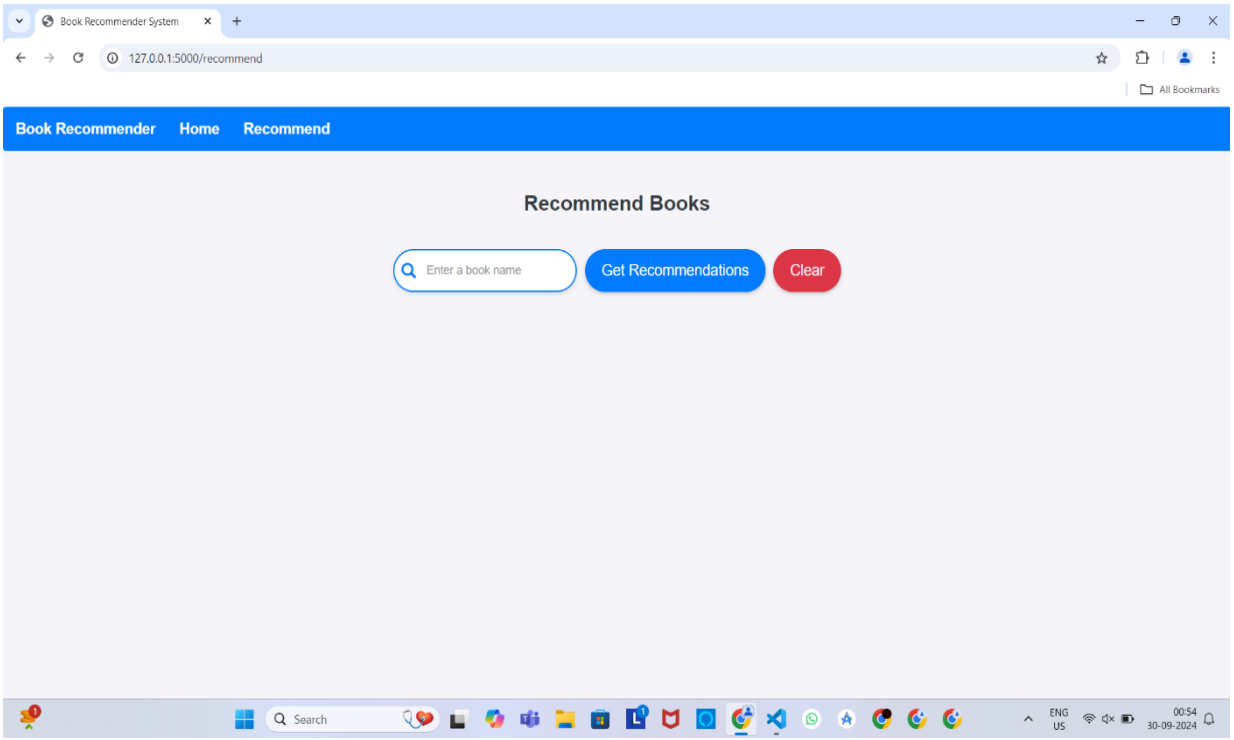
2. **Collaborative Filtering-Based Output:**
   - Once the user submits their chosen book title on the recommendation page, the system retrieves books that are most similar to the input title based on collaborative filtering. The results are presented in a similar card format to the homepage, but the content is tailored specifically to the user's input.
   - **Example Output:**

     If the user enters "Harry Potter", the system might return recommendations such as "Percy Jackson", "The Chronicles of Narnia", and other books with similar themes or audience preferences. Each recommendation includes the book's cover image, title, and author, providing a visually consistent and intuitive experience.

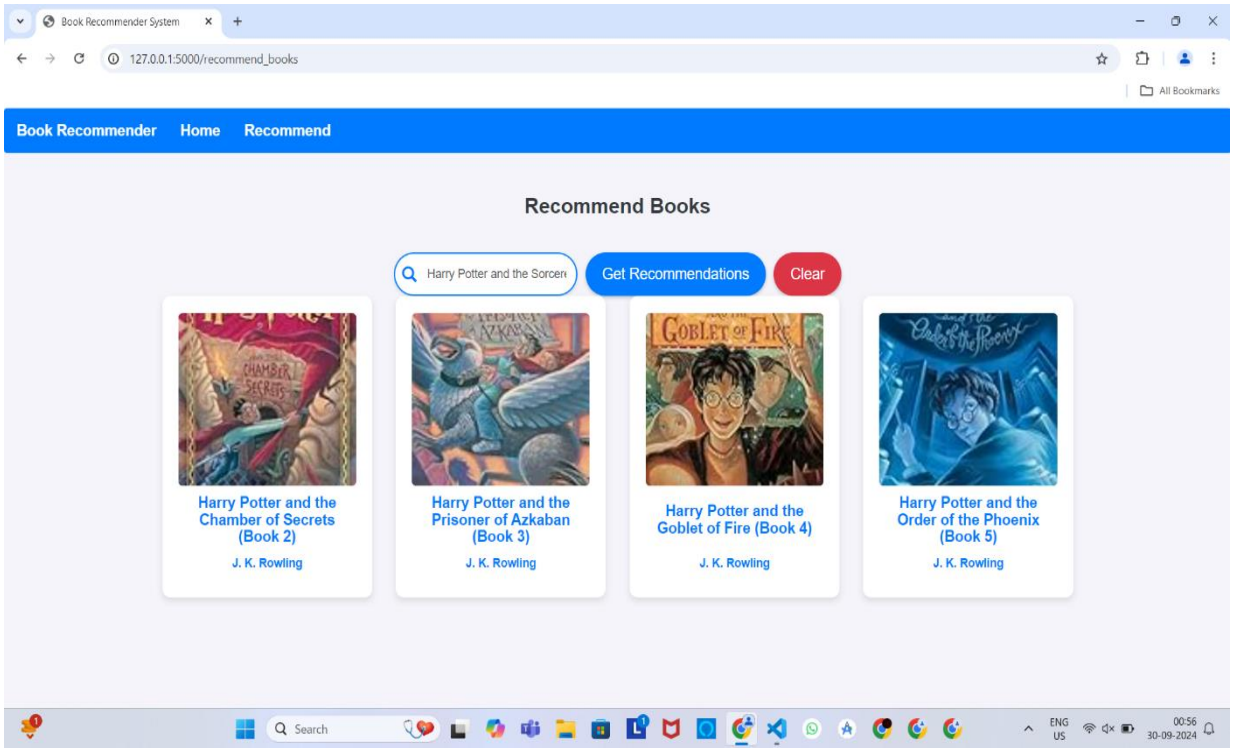
1) Initial view of Top Books:



2) Get recommendation page:

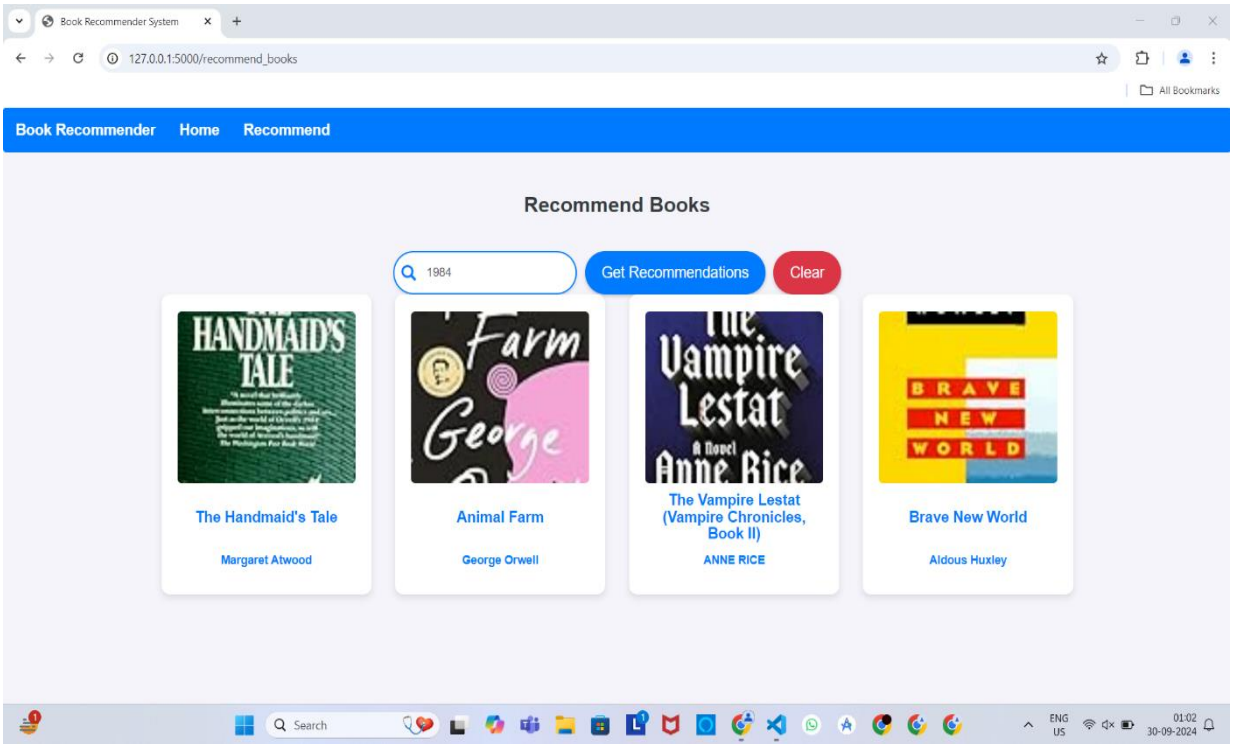


3) Recommended results:



4) Recommended results:

:



B.3 Observations and learning:

TERNA ENGINEERING COLLEGE

DEPARTMENT OF COPUTER ENGINEERING

"Book Recommendation System"

BE/SEM-VII

Anjali Shinde. [C-22] Student ID –TU3F2122159

Harsh Minde. [C-28] Student ID -TU3F2122164

Rushikesh Jadhav. [C-31] Student ID -TU3F2122167

Under the guidance of

Dr. Siddharth Hariharan.

Academic Year

2024-25

CONTENT

1. Introduction

As the digital age progresses, the sheer volume of content available to users - books, music, movies has grown exponentially. Consequently, recommendation systems have become a critical aspect of web applications, assisting users in finding the most relevant and personalized content

The Book Recommendation System is designed to simplify the process of book discovery by providing personalized recommendations to users based on their past interactions or the popularity of books within the system. This project focuses on using Natural Language Processing (NLP) techniques in conjunction with machine learning algorithms to create a highly effective recommendation model.

2. Problem Statement

Readers often find it difficult to select new books due to the overwhelming volume of choices available. This project addresses the challenge by implementing a Book Recommendation System that employs two main strategies:

1. Popularity-Based Recommendations: Highlights top -rated books based on user feedback.
2. Collaborative Filtering: Recommends books based on a user's interaction with other similar books and the preferences of other users.

3. Objective

The main objective of this project is to:

1. Develop a web-based system that allows users to search for books and receive personalized recommendations.
2. Implement two different recommendation techniques popularity-based and collaborative filtering to cater to different user preferences.
3. Create a user-friendly Graphical User Interface (GUI) that displays the top 50 books and provides recommendations in real-time.

4. Scope Of The Project

This project is designed as a scalable web application capable of handling a growing user base and dataset. The system supports both popularity-based and personalized recommendations using collaborative filtering techniques. It is built using Flask for the backend, Bootstrap for the frontend, and Python libraries for machine learning.

5. Software & hardware Requirement

1. **Operating System** : Windows 10/11 or Linux (Ubuntu 20.04 or above).
2. **Programming Language** : Python 3.x (latest version recommended).
3. **Integrated Development Environment (IDE)** : Visual Studio Code or PyCharm.
4. **Web Framework** : Flask (micro web framework written in Python).
5. **Libraries Used** : **5.1 Flask**: To manage the web server and routing.
   5.2 Numpy: For handling arrays and mathematical operations.
   5.3 Pandas: For managing and processing data in tabular form.
   5.4 Pickle: For serializing and de-serializing Python objects (used to store the model).
   5.5 Scikit-learn: For implementing the machine learning algorithms (Collaborative Filtering).
6. **Frontend Technologies** : HTML5, CSS3, Bootstrap 3.3.7.
7. **Databases**: CSV files containing books, users, and ratings data.

6. Software Design

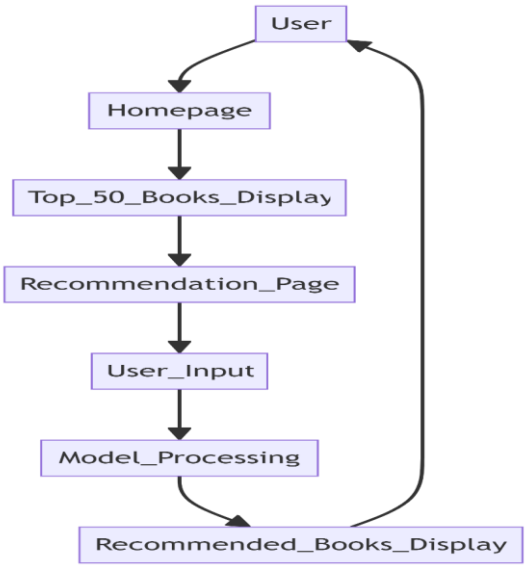


Fig: 6.1

6. Software Design

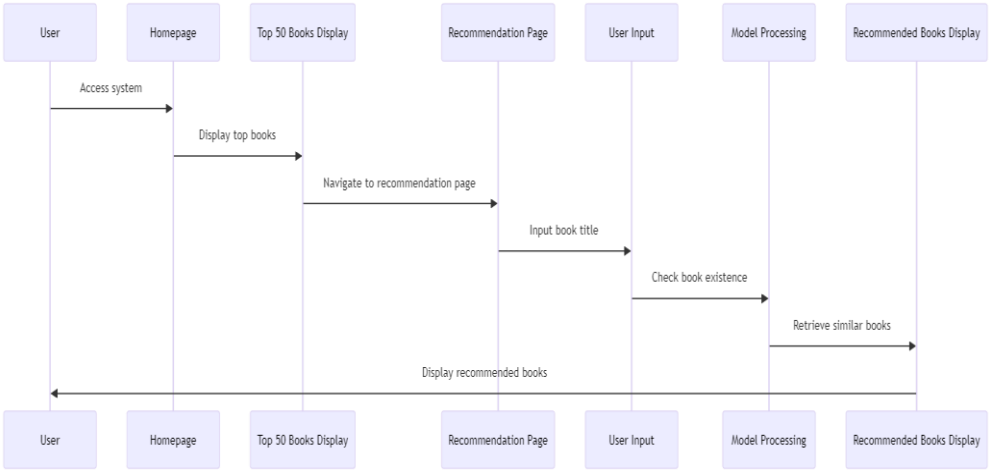


Fig: 6.2

7. Algorithm & Method Used

7.1 Popularity-Based Recommender System:

This method ranks books based on their popularity metrics, specifically the **number of ratings** and the **average rating** for each book. The implementation follows these steps:

1. **Data Aggregation** : Ratings data is aggregated to count the number of ratings and calculate the average rating for each book.
2. **Filtering**: Only books with a minimum of 250 ratings are considered.
3. **Top 50 Selection** : The books are then sorted based on their average rating, and the top 50 books are selected for display.

7. Algorithm & Method Used

7.1 Popularity-Based Recommender System:

Algorithmic Flow :

1. Aggregate rating data.
2. Filter out books with fewer than 250 ratings.
3. Sort books by average rating.
4. Select the top 50 books for display.

7. Algorithm & Method Used

7.2 Collaborative Filtering Based Recommender System:

Collaborative filtering is a method used to recommend items (books) to a user by finding similarities between items based on user ratings. The **cosine similarity** method is used to calculate the similarity between books based on user ratings.

Steps in Collaborative Filtering :

1. **Pivot Table Creation** : A pivot table is created where rows represent book titles, columns represent user IDs, and cells contain the rating a user has given to a book.
2. **Cosine Similarity Calculation** : Using the **cosine similarity** metric, the similarity between books is calculated.
3. **Recommendation Retrieval** : When a user inputs a book, the system identifies similar books based on their similarity scores and presents them as recommendations.

7. Algorithm & Method Used

7.2 Collaborative Filtering Based Recommender System:

Algorithmic Flow:

1. Create a pivot table using books and users.
2. Compute cosine similarity between books.
3. Identify top similar books based on the input book.
4. Present recommendations.

8. Project Implementation

8.1 Data Collection and Preprocessing:

Datasets Used:

1. **Books Dataset**: Contains details about books including title, author, and image URL.
2. **Users Dataset**: Contains information about users, primarily used for collaborative filtering.
3. **Ratings Dataset**: Contains user ratings for different books. This dataset forms the basis for both the popularity-based and collaborative filtering models.

Preprocessing Steps:

1. **Data Cleaning**: Handle missing values and remove any duplicates in the data.
2. **Merging Datasets**: Combine the books and ratings data into a single dataset that can be used for modeling.
3. **Data Splitting**: Split the data into training and test sets for evaluation (optional if further model training is needed).

8. Project Implementation

8.2 Model Training and Serialization:

The models are trained using the preprocessed datasets:

1. **Popularity Model**: This model is trained by calculating the number of ratings and the average rating for each book. The top 50 books are then serialized using **Pickle** for fast access during runtime.
2. **Collaborative Filtering Model** : The pivot table is created, and cosine similarity scores are calculated for every book. The similarity matrix is then serialized for efficient retrieval of recommendations.

Pickle Files:

1. **popular.pkl:** Contains the top 50 popular books.
2. **similarity_scores.pkl :** Stores the cosine similarity scores between books.
3. **pt.pkl:** Contains the pivot table used for collaborative filtering.

8. Project Implementation

8.3 Flask Application:

The **Flask App** serves as the backbone of the web application, handling user interactions and routing requests:

1. **Homepage Route (/)** : Displays the top 50 books using the popularity -based model.
2. **Recommendation Page Route (/recommend)** : Handles the input of book names and retrieves recommendations from the collaborative filtering model.

9. Conclusion

The **Book Recommendation System** successfully implements a two-fold recommendation strategy that includes both **popularity-based** and **collaborative filtering** techniques. The system provides users with a simple and intuitive interface that enables them to discover popular books or find recommendations based on their specific input.

By leveraging **Natural Language Processing (NLP)** and **machine learning** techniques, this project demonstrates the power of recommendation systems in improving user experience on web platforms. The use of **Flask** for web application development and **Python** for machine learning provides a robust, scalable solution that can be extended with additional features such as user profiles, real-time recommendations, and personalized recommendation models.

10. Reference

1. https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset
2. Kaggle
3. Youtube

B.4 Conclusion:

The **Book Recommendation System** effectively implements a dual recommendation strategy, combining popularity-based recommendations and collaborative filtering techniques. By utilizing **Natural Language Processing (NLP)** and **machine learning**, the system demonstrates the potential of modern recommendation systems to significantly improve user experiences on web platforms. The simplicity and intuitiveness of the user interface allow users to easily discover popular books or obtain personalized recommendations based on their specific input.

Key aspects of the system include:

- **User Interaction**:

  The system enables seamless interaction, where users input book titles and receive tailored recommendations. Flask handles these interactions efficiently, ensuring the system remains responsive even under different user requests.

- **Efficient Computation**:

  By precomputing models and leveraging serialized files for data storage, the system quickly processes large datasets and delivers recommendations without unnecessary delays, ensuring a smooth user experience.

- **Engaging User Experience**:

  Balancing static, popularity-based suggestions with dynamic, collaborative filtering results, the system effectively caters to a wide range of user preferences, enhancing engagement and satisfaction.

The project showcases the integration of **Flask** for web application development and **Python** for machine learning, creating a scalable and robust solution. Additionally, the system provides opportunities for future extensions, such as adding user profiles, real-time recommendation updates, and advanced personalization techniques. Through a successful combination of web development and data science, this recommendation platform delivers both functional and personalized results, making it a powerful tool for book discovery and recommendation.

THANK YOU !!