Book Recommendation System

Mini Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Engineering (Computer Engineering)

by:

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.

Department of Computer Engineering

TERNA ENGINEERING COLLEGE

Nerul (W), Navi Mumbai 400706

(University of Mumbai)

(2024-2025)

Internal Approval Sheet

TERNA ENGINEERING COLLEGE, NERUL

Department of Computer Engineering

Academic Year 2024-2025

CERTIFICATE

This is to certify that project entitled “Book Recommendation System” is bonafide workof:

Anjali Shinde. [C-22]	Student ID –TU3F2122159
Harsh Minde. [C-28]	Student ID -TU3F2122164
Rushikesh Jadhav. [C-31]	Student ID -TU3F2122167

Submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the Bachelor of Engineering (Computer Engineering).

Guide **Head of Department** **Principal**

Table of Contents

Chapter 1
Introduction

1.1 Motivation :

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.

1.2 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.

1.3 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.

1.4 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.

Chapter 2

Software & Hardware Requirements

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.

Chapter 3

Software Design

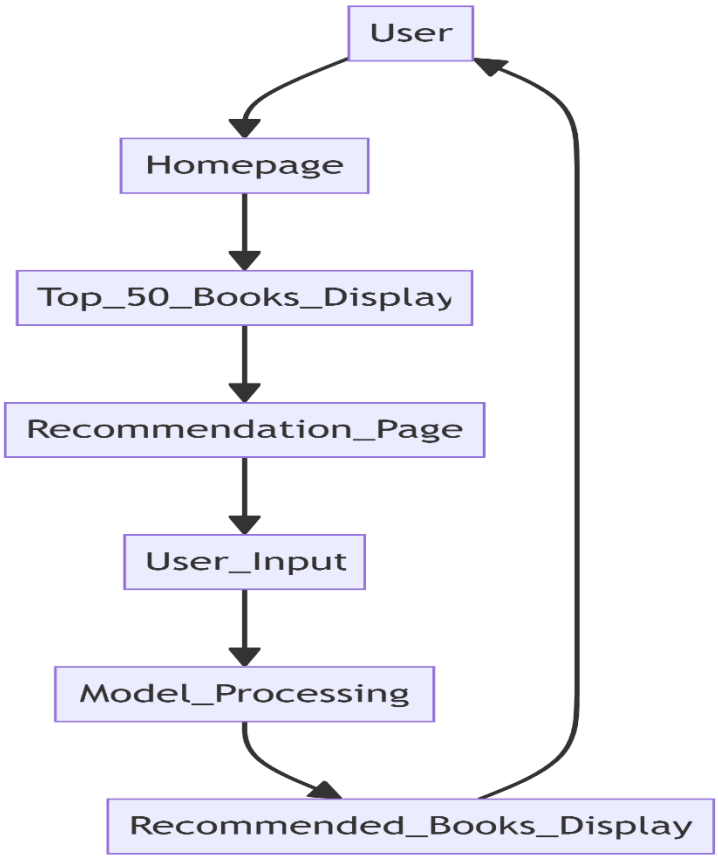

Fig: 3.1

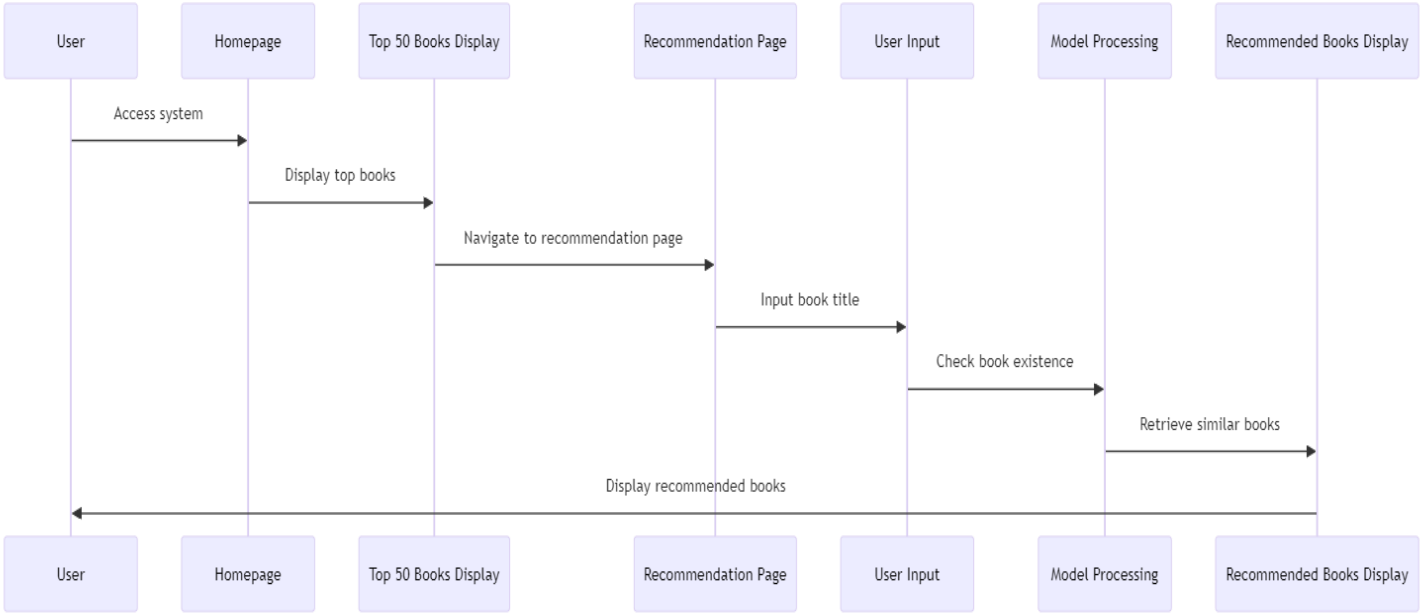

Fig: 3.2

Chapter 4

Algorithms & Methods Used

4.1 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.

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.

4.2 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.

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.

Chapter 5

Project Implementation

5.1 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).

5.2 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.

5.3 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.

Chapter 6

Program

Input Code:

App.py:

```
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')
    index = np.where(pt.index == user_input)[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)
        print(data)

    return render_template('recommend.html',data=data)
    if __name__ == '__main__':
         app.run(debug=True)
```

index.html:

```
<!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;
    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;
}
```

```
        .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>
            <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>
            </div>
            {% endfor %}
        </div>
    </div>

</body>
</html>
```

recommend.html:

```
<!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;
    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 {
    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 */
    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 */
.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>
        </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) {
        const userInput = document.getElementById('bookInput').value.trim();
        if (!userInput) {
            event.preventDefault();
            document.getElementById('errorMessage').style.display = 'block';
        }
    });
```

```
        document.getElementById('clearBtn').addEventListener('click', function() {
            // Clear the input field
            document.getElementById('bookInput').value = '';

            // Hide the error message if it exists
            const errorMessage = document.getElementById('errorMessage');
            if (errorMessage) {
                errorMessage.style.display = 'none'; // Hide error message
            }

            // Clear the displayed results
            const rowDiv = document.querySelector('.row');
            if (rowDiv) {
                rowDiv.innerHTML = ''; // Clear all contents of the row
            }
        });

        // If no data is found, display the 'noResultsMessage'
        {% if not data %}
        document.getElementById('noResultsMessage').style.display = 'block';
        {% endif %}
    </script>

</body>
</html>
```

Chapter 7

Output Snapshots

1) Initial view of Top Books:

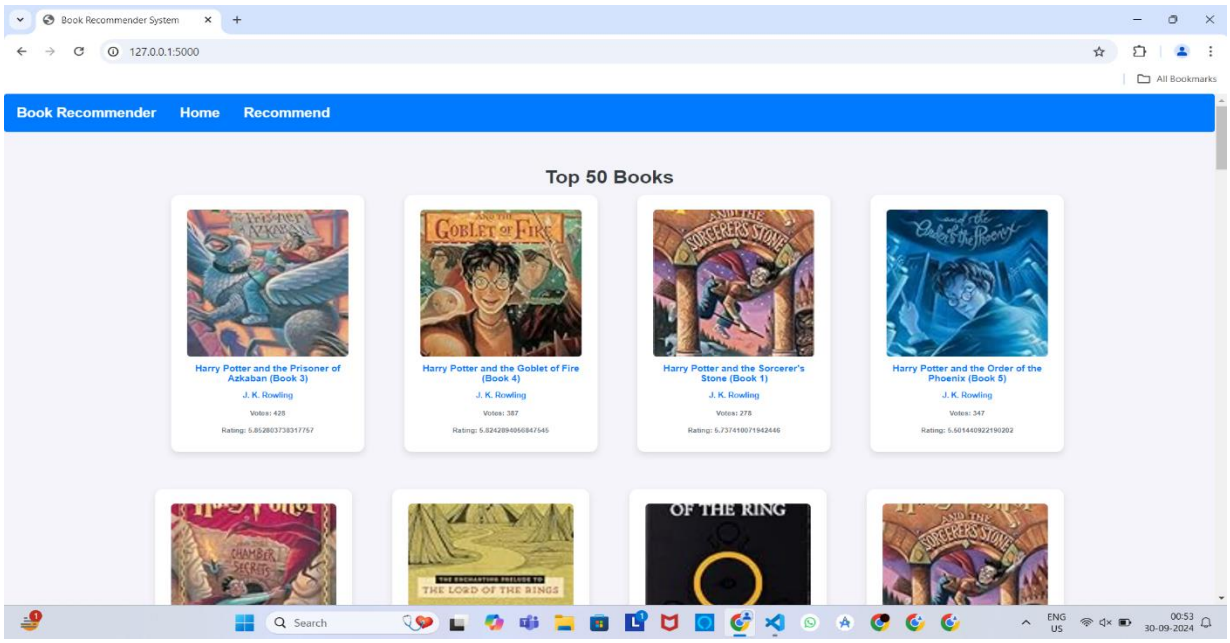

Fig: 7.1

2) Get recommendation page:

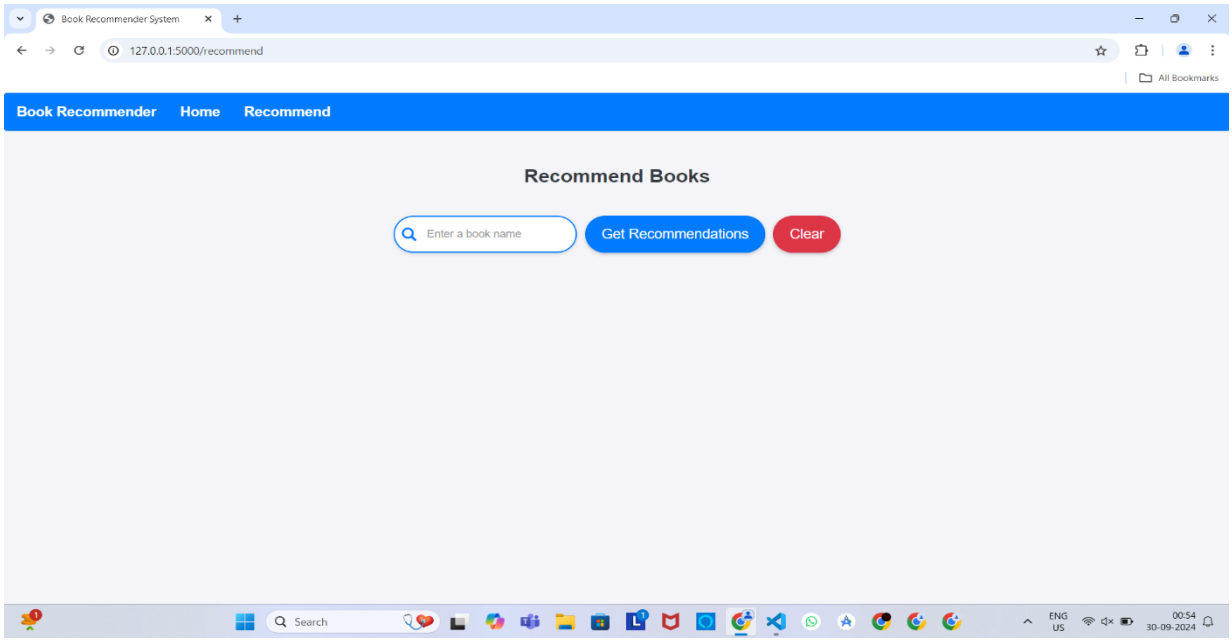

Fig: 7.2

3) Recommended results:

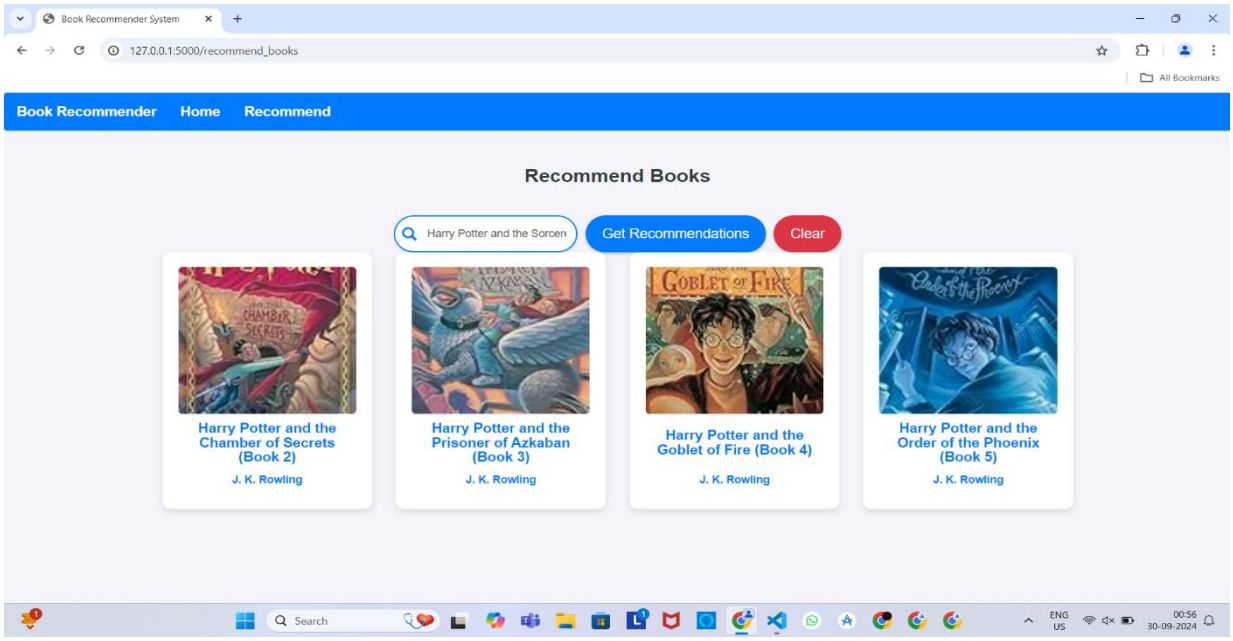

Fig: 7.3

4) Recommended results:

:

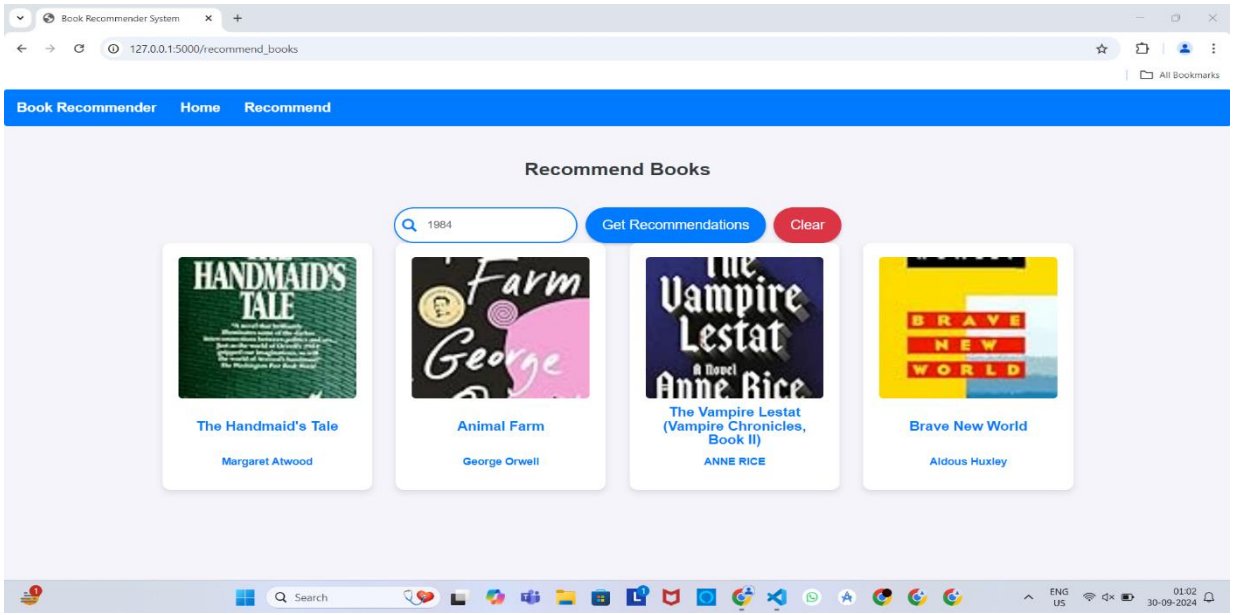

Fig: 7.4

Chapter 8
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.

Chapter 9

Reference

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

THANK YOU !