RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BOOK RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

Submitted by

PRAGADEESH KUMAR L D	**231501117**
RAGHAVENDHRA M K	**231501115**
NITHESH KUMAR S	**231501114**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report **“BOOK RECOMMENDATION SYSTEM”** is the bonafide work of **“PRAGADEESH KUMAR L D (231501117) , RAGHAVENDHRA M K (231501125), NITHESH KUMAR S (231501114)”** who carried out the project work under my supervision.

Submitted for the Practical Examination held on ____________________________

SIGNATURE

Mr. U. Kumaran,
Assistant Professor (SS)
AIML,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Book Recommendation System is a cutting-edge digital platform designed to transform how readers discover and engage with books. By leveraging advanced algorithms and user preferences, the system provides personalized book recommendations tailored to individual tastes, reading habits, and interests. This intuitive platform not only simplifies the book discovery process but also enables users to explore diverse genres, authors, and titles, ensuring a curated experience that enhances their reading journey.

With integrated features for tracking reading progress, creating wishlists, and sharing reviews, the Book Recommendation System fosters a dynamic community of readers and promotes an interactive exchange of literary insights. By analyzing user behavior and book trends, the platform offers highly accurate suggestions, helping readers discover books they might otherwise overlook.

Beyond just offering book recommendations, the system encourages deeper engagement with literature, promoting diverse reading experiences and expanding literary horizons. Future upgrades, such as social reading features, audiobook integration, and AI-driven content personalization, will further refine and enhance the platform's capabilities. The *Book Recommendation System* is set to redefine how people discover and connect with books, making it easier than ever to find the next great read while fostering a more informed and engaged reading community.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

2. SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES

2.2.1 SQL

2.2.2 PYTHON

3. REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

3.5 NORMALIZATION

4. PROGRAM CODE

5. PROJECT SCREENSHOTS

6. CONCLUSION

7. REFERENCE

1. INTRODUCTION

1.1 Introduction

This project is a Book Recommendation System, designed to provide users with an interactive and efficient way to search for and explore books based on various criteria, such as title, author, genre, and ratings. The system leverages a database of books, allowing users to filter and sort through a collection of titles to find those that match their preferences.

The recommendation system is implemented as a web application using a combination of frontend and backend technologies:

- Frontend: HTML, CSS, and JavaScript provide a user-friendly interface where users can easily search, sort, and view book details.
- Backend: Flask, a Python-based web framework, serves as the backend, connecting to a MySQL database where book data is stored and retrieved.
- Database: MySQL is used to manage book data, supporting search and sorting queries efficiently.

This application demonstrates key concepts in web development, including client-server communication, data management, and real-time data retrieval based on user queries.

1.2 Objectives

The primary objectives of this Book Recommendation System are:

1. Efficient Book Retrieval: Enable users to search for books by title, author, or genre and retrieve results quickly.
2. Interactive Sorting and Filtering: Provide options to sort books based on criteria like average rating, number of ratings, or title, allowing users to personalize their browsing experience.
3. Data Management and Storage: Use a structured database to store, retrieve, and manage book information securely.
4. User-Friendly Interface: Develop an intuitive interface that facilitates seamless interaction, where users can effortlessly search and view books.

5. Scalable Architecture: Structure the backend and database to support future expansion, such as adding more books, user accounts, or enhanced recommendation algorithms.

1.3 Modules

This system is divided into several modules, each handling a specific part of the application functionality:

1.3.1 Frontend Module

- Purpose: Provides the user interface where users can search, filter, and view book recommendations.
- Components:
  - Search and Filter Interface: An input field allows users to search for books by title, author, or genre. Dropdown menus enable sorting options, which are reflected in real-time as the search is conducted.
  - Dynamic Book Display: Uses JavaScript to fetch book data from the backend and render it on the page, including details like the title, author, description, genres, average rating, and number of ratings.

1.3.2 Backend Module

- Purpose: Acts as the intermediary between the frontend and the database, managing requests, processing data, and returning the appropriate results.
- Components:
  - Flask Server: Handles HTTP requests from the frontend. Includes routes for displaying the home page and fetching book data from the database.
  - API Endpoint (/books): Accepts parameters for search and sorting, interacts with the database, and returns filtered and sorted book data as JSON.
  - Data Processing: Incorporates a helper function to apply search and sorting criteria dynamically based on user input.

1.3.3 Database Module

- Purpose: Stores and organizes book data, allowing for efficient querying and management.
- Components:
    - MySQL Database: The database stores the book information, with fields for title, author, description, genres, average rating, number of ratings, and URL.
    - Schema Setup: SQL schema script creates the required database structure, ensuring all relevant fields are included and properly indexed.
    - Data Import Script (import_csv_to_db.py): Reads book data from a CSV file and populates the database, which can be run periodically or as needed to update the book collection.

1.3.4 Models Module

- Purpose: Defines the data structure for each book entry, making it easier to handle and manage book information in the backend.
- Components:
    - Book Class: A Python class that encapsulates all properties of a book (title, author, description, genres, average rating, number of ratings, and URL), enabling structured representation and easy data manipulation.

2. SURVEY OF TECHNOLOGIES

2.1 Software Description

The Book Recommendation System leverages a combination of web technologies and software tools to create a full-stack web application with a dynamic user interface, server-side processing, and a structured database. The key components include:

- **Flask**: Flask is a micro web framework written in Python, used here to build the backend server that handles client requests, processes search and filter queries, and communicates with the MySQL database. Flask's lightweight nature makes it a great choice for applications requiring high flexibility and customizability.
- **MySQL**: MySQL is a widely-used relational database management system that stores structured book data in a database. It is highly reliable for handling large data volumes and supports SQL querying, which allows for complex searches and sorting operations on book records.
- **MySQL Connector/Python**: This library allows the Python application to interact with the MySQL database, enabling efficient data insertion, querying, and retrieval directly from Python scripts. It handles authentication, connection pooling, and query execution, making it a key tool in the backend.
- **HTML/CSS/JavaScript**: HTML, CSS, and JavaScript are the core technologies for the frontend, providing the structure, styling, and interactivity for the user interface. The frontend allows users to search, filter, and view books in a user-friendly format.

2.2 Languages

The core languages used in the development of this application are SQL and Python, each playing a crucial role in different parts of the system.

2.2.1 SQL

SQL (Structured Query Language) is a standard programming language used for managing and manipulating relational databases. In this project, SQL is primarily used in the following ways:

- **Database Creation and Management**: SQL commands are used to create the book_recommendation database and the books table, as defined in the schema.sql script. SQL statements like CREATE TABLE, INSERT, SELECT, and UPDATE define the database structure, manage entries, and modify data as needed.
- **Data Insertion**: SQL is used to insert book data from CSV files into the database. This is done using the import_csv_to_db.py script, where SQL INSERT statements add each book's details (title, author, genres, etc.) to the books table.
- **Data Querying**: SQL SELECT statements retrieve book records from the database based on user search and sorting preferences. These queries, generated dynamically in the backend, filter and sort data based on fields like title, author, and average rating, providing only the relevant records to the user.

Advantages of SQL:

- **Structured Data Management**: SQL is ideal for managing structured data with defined relationships, which makes it well-suited for this project where each book record has specific fields.
- **Efficient Querying**: SQL provides powerful capabilities for filtering, sorting, and aggregating data, allowing for quick response times in applications with large datasets.

2.2.2 Python

Python is an interpreted, high-level, and versatile programming language, known for its readability and robust libraries. Python is the primary language for the backend in this project, performing tasks like handling HTTP requests, processing data, and interacting with the database.

- **Flask Framework**: Python is used with the Flask framework to build the backend server. Flask's routing, request handling, and template rendering capabilities make it straightforward to develop RESTful APIs that support search and sorting features for books.
- **MySQL Connector/Python Library**: Python's mysql.connector library allows secure and efficient interactions with the MySQL database. The get_db_connection

function in database.py uses this library to establish a connection with the database and execute queries.

- **CSV Handling**: Python's built-in csv module reads and processes book data from CSV files, which is then inserted into the database. This is essential for initially populating the database with a large dataset of books.

Advantages of Python:

- **Rapid Development**: Python's simplicity and extensive libraries reduce development time, allowing for quick implementation and testing.
- **Strong Database Integration**: Python libraries like mysql.connector enable seamless communication with databases, making it efficient to handle and manage large datasets.
- **Versatile Frameworks**: Flask, a lightweight framework, provides the flexibility needed for a customized backend, while other Python libraries support everything from data handling to web development.

3. REQUIREMENTS AND ANALYSIS

3.1 Requirement Specification

The **Book Recommendation System** is designed to meet the following functional and non-functional requirements:

Functional Requirements

1. **User Search**: The system should allow users to search for books based on title, author, or genre.
2. **Filtering and Sorting**: Users can filter and sort books by fields like average rating, number of ratings, or title.
3. **Database Interaction**: The backend must support retrieving, inserting, and managing book data in the MySQL database.
4. **User Interface**: The frontend should present an intuitive layout that displays book details and provides search and sorting options.
5. **Data Import**: The system should support bulk importing of book data from a CSV file.

Non-Functional Requirements

1. **Performance**: The system should handle queries efficiently to maintain a responsive user experience, even with a large dataset.
2. **Scalability**: The architecture should allow for easy expansion, including the addition of more book records or future features.
3. **Usability**: The interface should be user-friendly and accessible, providing clear search, filter, and display functions.
4. **Security**: Database connections should be secure, and the application should validate user inputs to prevent SQL injection and other vulnerabilities.

3.2 Hardware and Software Requirements

Hardware Requirements

- **Server/Computer**: A standard server or PC with at least 4GB RAM and 2.5 GHz processor.
- **Storage**: At least 500MB of disk space to store the book dataset and other required files.

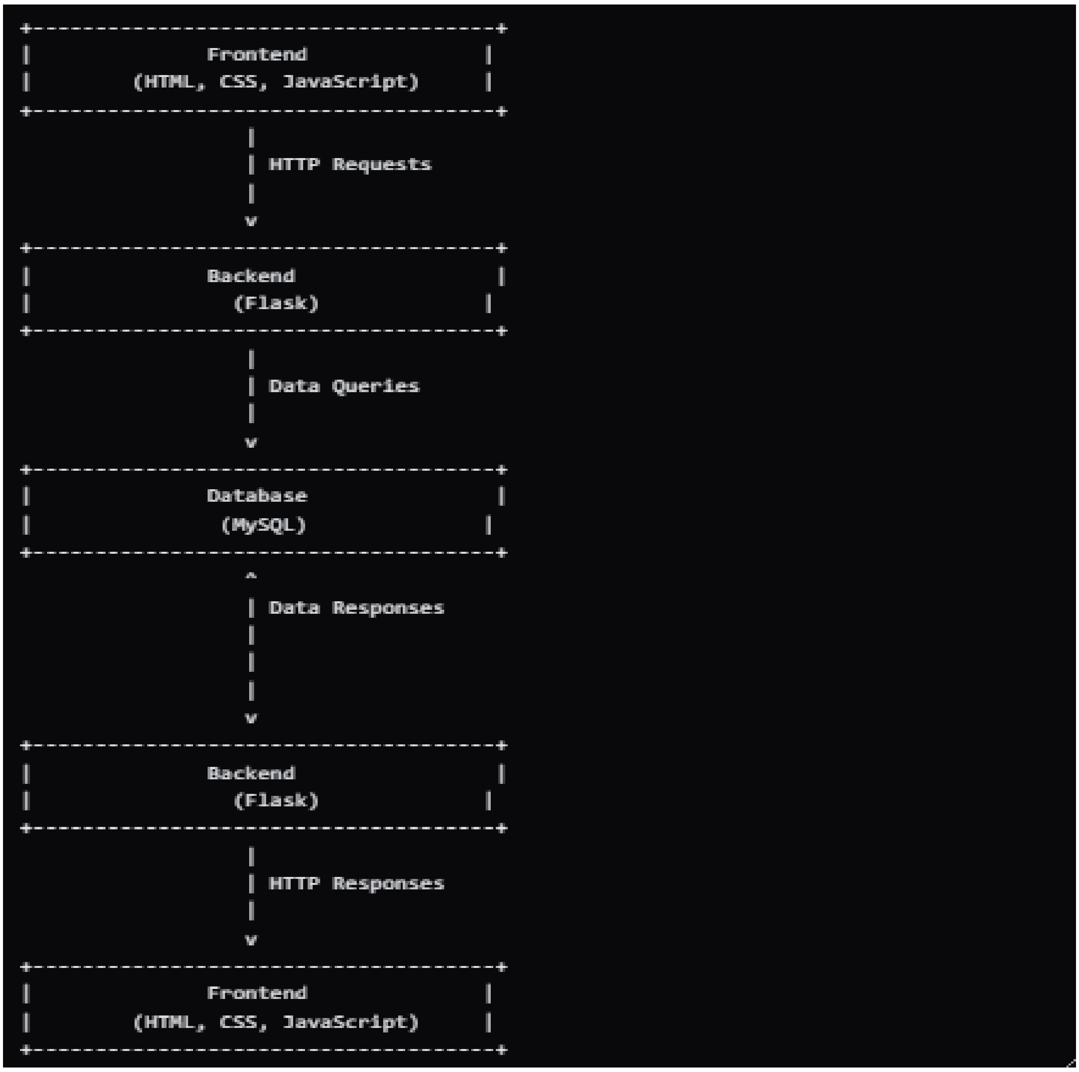
Software Requirements

- **Operating System**: Windows, macOS, or any Linux distribution.
- **Database Management System**: MySQL, for managing the book data.
- **Backend Framework**: Flask, running on Python 3.x, for server-side programming.
- **Frontend Technologies**: HTML, CSS, and JavaScript for the user interface.
- **Python Libraries**: mysql.connector for database interaction and flask_cors for handling cross-origin requests.
- **CSV File for Data**: A CSV file containing book data (title, author, genre, ratings, etc.) to be imported into the database.

3.3 Architecture Diagram

An architecture diagram provides an overview of how the components of the system interact. Here's a description of the architecture:

- **Frontend (Client-Side)**: The user interacts with the frontend, which is built using HTML, CSS, and JavaScript. The frontend makes requests to the Flask server to retrieve book data.
- **Backend (Server-Side)**: Flask acts as the web server, handling HTTP requests from the frontend. It processes these requests and communicates with the MySQL database as needed.
- **Database**: The MySQL database stores all book-related data, structured for efficient retrieval. The backend retrieves and manages this data based on user queries.



```
+--------------------------------------+
|              Frontend                |
|        (HTML, CSS, JavaScript)       |
+--------------------------------------+
                  |
                  | HTTP Requests
                  |
                  v
+--------------------------------------+
|              Backend                 |
|               (Flask)                |
+--------------------------------------+
                  |
                  | Data Queries
                  |
                  v
+--------------------------------------+
|              Database                |
|              (MySQL)                 |
+--------------------------------------+
                  ^
                  | Data Responses
                  |
                  |
                  |
                  v
+--------------------------------------+
|              Backend                 |
|               (Flask)                |
+--------------------------------------+
                  |
                  | HTTP Responses
                  |
                  v
+--------------------------------------+
|              Frontend                |
|        (HTML, CSS, JavaScript)       |
+--------------------------------------+
```

3.4 ER Diagram

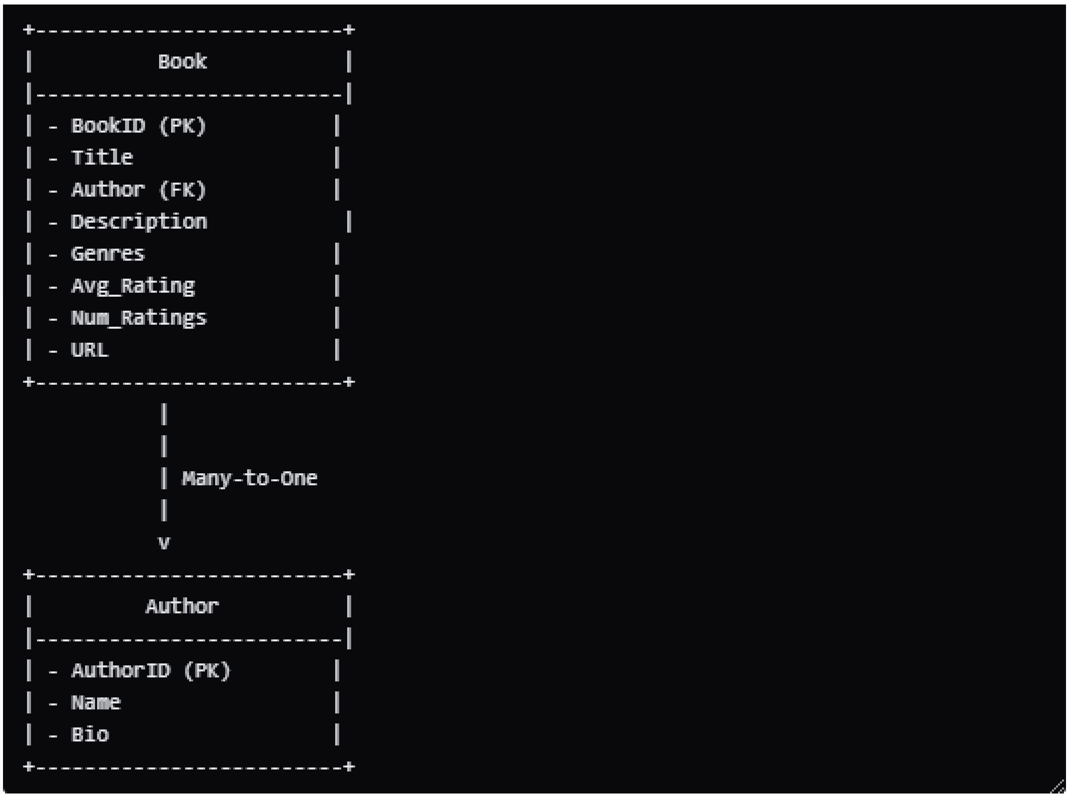
An Entity-Relationship (ER) diagram provides a visual representation of the data model for the Book Recommendation System. The following entities and relationships could be included:

1. **Entities**:
   - **Book**: Represents each book in the collection.
   - **Author** (optional if additional author details are needed beyond just a name).
2. **Attributes**:

- **Book**: BookID (PK), Title, Author, Description, Genres, Avg_Rating, Num_Ratings, URL.

3. **Relationships**:
   - If you add an Author entity, the **Book** entity would have a **Many-to-One** relationship with the **Author** entity.

(An ER diagram tool can help visualize the Book entity with fields and any potential relationships.)



```
+------------------------+
|          Book          |
|------------------------|
| - BookID (PK)         |
| - Title               |
| - Author (FK)         |
| - Description          |
| - Genres              |
| - Avg_Rating          |
| - Num_Ratings         |
| - URL                 |
+------------------------+
           |
           |
           | Many-to-One
           |
           v
+------------------------+
|         Author         |
|------------------------|
| - AuthorID (PK)       |
| - Name                |
| - Bio                 |
+------------------------+
```

3.5 Normalization

Normalization is the process of organizing the database to reduce redundancy and dependency, ensuring efficient data storage and retrieval.

1. **1NF (First Normal Form)**: Ensures that all data is stored in tables with atomic values. Each field in the books table, like Title, Author, Genres, etc., holds a single value per row. In cases where books have multiple genres, genres could either be stored as a comma-separated string or normalized further by creating a separate **Genre** table.
2. **2NF (Second Normal Form)**: Achieved by ensuring that each non-primary key attribute is fully dependent on the primary key. Since each book is uniquely identified by BookID or Title (if unique), there are no partial dependencies in the books table.
3. **3NF (Third Normal Form)**: Achieved by ensuring that non-key attributes are not dependent on other non-key attributes. In this case, all book information like Description, Genres, Avg_Rating, etc., directly relates to the BookID and is independent of each other, satisfying 3NF.

4. PROGRAM CODE

4.1 Overview of Code Structure

The **Book Recommendation System** codebase is structured into several key files, each handling a specific function within the application. The main components include the backend server code, database connection scripts, models, data import scripts, and frontend files. Below is a breakdown of each file and its role in the project:

1. **app.py**: This is the main application file that configures the Flask server and defines routes for serving book data.
2. **database.py**: Contains the logic for establishing a connection to the MySQL database.
3. **import_csv_to_db.py**: This script imports data from a CSV file into the database, making it accessible to the application.
4. **models.py**: Defines the Book model to represent books and organize book-related data.
5. **schema.sql**: A SQL script for creating the database schema, defining tables and fields necessary for storing book data.
6. **index.html**: The main frontend file, displaying the book list, search, and sort options to the user.

4.2 Code Walkthrough

Here is a detailed explanation of each file, with key code snippets and functionality descriptions.

4.2.1 app.py

```
from flask import Flask, jsonify, render_template, request
from flask_cors import CORS
from database import get_db_connection
from models import Book


app = Flask(__name__)
CORS(app)
```

```
# Helper function to apply search and sorting
def apply_filters_and_sorting(query, params):
    search_term = params.get('search', None)
    sort_by = params.get('sort_by', 'avg_rating')
    sort_order = params.get('order', 'asc')

    # Apply search filter if search term is provided
    if search_term:
        query += " WHERE title LIKE %s OR author LIKE %s OR genres LIKE %s"
        # Use params as a list instead of a dict to avoid the AttributeError
        params_list = [f'%{search_term}%', f'%{search_term}%', f'%{search_term}%']
    else:
        params_list = []

    # Apply sorting
    if sort_by and sort_order:
        query += f" ORDER BY {sort_by} {sort_order.upper()}"

    # Return the query and the list of parameters for executing the query
    return query, params_list

@app.route('/')
def home():
    return render_template('index.html')

# Check that the API returns all necessary fields
@app.route('/books', methods=['GET'])
def get_books():
    connection = get_db_connection()
    cursor = connection.cursor(dictionary=True)
```

```
# Base query to select all books
query = 'SELECT * FROM books'
params = []

# Apply search and sorting based on URL query parameters
query, params = apply_filters_and_sorting(query, request.args.to_dict())

# Pagination: Handle 'page' and 'page_size' parameters
page = int(request.args.get('page', 1))
page_size = int(request.args.get('page_size', 10))
offset = (page - 1) * page_size
query += f" LIMIT {page_size} OFFSET {offset}"

cursor.execute(query, params)
books_data = cursor.fetchall()

# Get total number of books for pagination
cursor.execute("SELECT COUNT(*) AS total FROM books")
total_books = cursor.fetchone()['total']

connection.close()

books = []
for book in books_data:
    books.append({
        'title': book['title'],
        'author': book['author'],
        'description': book['description'],
        'genres': book['genres'],
        'avg_rating': book['avg_rating'],
        'num_ratings': book['num_ratings'],
        'url': book['url']
        })

    return jsonify({'books': books, 'totalBooks': total_books})


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

Description: app.py serves as the main entry point for the application, handling HTTP requests, managing API endpoints, and rendering the frontend template.

4.2.2 database.py

```
import mysql.connector

def get_db_connection():
    connection = mysql.connector.connect(
        host='localhost',
        user='root',
        password='your_password',
        database='book_recommendation'
    )
    return connection
```

Description: database.py provides a function to connect to the MySQL database, which is used by the Flask app to retrieve and manage book data.

4.2.3 import_csv_to_db.py

```
import csv
import mysql.connector

def import_csv_to_db():
    with open('books.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        connection = mysql.connector.connect(
            host='localhost',
            user='root',
            password='your_password',
            database='book_recommendation'
        )
        cursor = connection.cursor()
        for row in reader:
            Num_Ratings = row['Num_Ratings']
            try:
                Num_Ratings = int(Num_Ratings)
            except ValueError:
                Num_Ratings = 0
            cursor.execute("""
                INSERT INTO books (title, author, description, genres, avg_rating, Num_Ratings, url)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
            """, (
                row['Title'], row['Author'], row['Description'],
                row['Genres'], row['Avg_Rating'], Num_Ratings, row['URL']
            ))
        connection.commit()
        connection.close()
        print("CSV data imported successfully!")

if __name__ == "__main__":
    import_csv_to_db()
```

Description: import_csv_to_db.py is a utility script to import book data from a CSV file into the database, useful for initializing the database with data.

4.2.4 models.py

```
class Book:
    def __init__(self, title, author, description, genres, avg_rating, num_ratings, url):
        self.title = title
        self.author = author
        self.description = description
        self.genres = genres
        self.avg_rating = avg_rating
        self.num_ratings = num_ratings
        self.url = url
```

Description: models.py defines the Book class, which represents a book and stores attributes like title, author, description, genres, rating, and URL.

4.2.5 schema.sql

```
-- Create the database if it doesn't exist
CREATE DATABASE IF NOT EXISTS book_recommendation;


-- Use the created database
USE book_recommendation;


-- Table to store books
CREATE TABLE IF NOT EXISTS books (
id INT AUTO_INCREMENT PRIMARY KEY,
title VARCHAR(255) NOT NULL,
author_id INT NOT NULL,
description TEXT,
avg_rating DOUBLE CHECK (avg_rating BETWEEN 0 AND 5),
num_ratings BIGINT UNSIGNED DEFAULT 0,
url VARCHAR(255),
published_date DATE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

FOREIGN KEY (author_id) REFERENCES authors(id) ON DELETE CASCADE

);


-- Table to store authors

CREATE TABLE IF NOT EXISTS authors (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

bio TEXT,

date_of_birth DATE,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP

);


-- Table to store genres

CREATE TABLE IF NOT EXISTS genres (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL UNIQUE,

description TEXT,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP

);


-- Table to establish many-to-many relationship between books and genres

CREATE TABLE IF NOT EXISTS book_genres (

book_id INT NOT NULL,

genre_id INT NOT NULL,

PRIMARY KEY (book_id, genre_id),

FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE,

FOREIGN KEY (genre_id) REFERENCES genres(id) ON DELETE CASCADE
```

```
);

-- Table to store users (if applicable, for recommendations or reviews)
CREATE TABLE IF NOT EXISTS users (
id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(50) NOT NULL UNIQUE,
email VARCHAR(100) NOT NULL UNIQUE,
password_hash VARCHAR(255) NOT NULL,
date_of_birth DATE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- Table to store reviews by users for books
CREATE TABLE IF NOT EXISTS reviews (
id INT AUTO_INCREMENT PRIMARY KEY,
book_id INT NOT NULL,
user_id INT NOT NULL,
rating INT CHECK (rating BETWEEN 1 AND 5),
review_text TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Table to store user-recommendations
CREATE TABLE IF NOT EXISTS recommendations (
id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT NOT NULL,
book_id INT NOT NULL,
recommended_on TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE
);

-- Optional: Add indexes for optimization
CREATE INDEX idx_title ON books(title);
CREATE INDEX idx_author ON authors(name);
CREATE INDEX idx_genre_name ON genres(name);
CREATE INDEX idx_user_email ON users(email););
```

Description: schema.sql defines the database schema for the book recommendation application, including fields for storing book information.

4.2.6 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 Recommendation System</title>
  <style>
    /* General Layout Styles */
    body {
      font-family: 'Arial', sans-serif;
      background-color: #f4f7fc;
      margin: 0;
      padding: 20px;
      color: #333;
    }
```

```
h1 {
   color: #5c6bc0;
   text-align: center;
   margin-bottom: 30px;
}

/* Search Container Styling */
.search-container {
   text-align: center;
   margin-bottom: 20px;
}

#searchTerm {
   padding: 10px;
   width: 50%;
   font-size: 16px;
   border-radius: 5px;
   border: 1px solid #ddd;
   transition: all 0.3s ease;
}

#searchTerm:focus {
   border-color: #5c6bc0;
   outline: none;
}

#sortBy, #sortOrder {
   padding: 10px;
   margin-left: 10px;
   font-size: 16px;
   border-radius: 5px;
   border: 1px solid #ddd;
```

```
    transition: all 0.3s ease;
}

#sortBy:hover, #sortOrder:hover {
    border-color: #5c6bc0;
}

/* Book List Styling */
.book-list {
    opacity: 0;
    transition: opacity 0.5s ease-in-out;
}

.book-list.loaded {
    opacity: 1;
}

.book-item {
    background-color: #fff;
    padding: 20px;
    margin: 15px 0;
    border-radius: 8px;
    border: 1px solid #ddd;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    transition: all 0.3s ease;
}

.book-item:hover {
    transform: translateY(-5px);
    box-shadow: 0 6px 8px rgba(0, 0, 0, 0.15);
    border-color: #5c6bc0;
}
```

```
.book-item h3 {
    color: #5c6bc0;
    font-size: 20px;
    margin-bottom: 10px;
}

.book-item p {
    font-size: 14px;
    margin: 5px 0;
}

.book-item a {
    color: #5c6bc0;
    text-decoration: none;
}

.book-item a:hover {
    text-decoration: underline;
}

/* Pagination Styling */
.pagination {
    display: flex;
    justify-content: center;
    margin-top: 20px;
}

.pagination button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
```

```
            border: 1px solid #ddd;
            background-color: #fff;
            margin: 0 5px;
            border-radius: 5px;
            transition: background-color 0.3s ease, transform 0.3s ease;
        }

        .pagination button:hover {
            background-color: #5c6bc0;
            color: white;
            transform: translateY(-2px);
        }

        .pagination button:disabled {
            background-color: #f0f0f0;
            cursor: not-allowed;
        }

        .pagination button:disabled:hover {
            background-color: #f0f0f0;
        }
    </style>
</head>
<body>

    <h1>Book Recommendation System</h1>

    <!-- Search and Sorting Form -->
    <div class="search-container">
        <input type="text" id="searchTerm" placeholder="Search by title, author, or genre"
oninput="fetchBooks()">
```

```
        <select id="sortBy" onchange="fetchBooks()">
            <option value="avg_rating">Sort by Avg Rating</option>
            <option value="num_ratings">Sort by Num Ratings</option>
            <option value="title">Sort by Title</option>
        </select>

        <select id="sortOrder" onchange="fetchBooks()">
            <option value="asc">Ascending</option>
            <option value="desc">Descending</option>
        </select>
    </div>

    <!-- List of books -->
    <ul id="bookList" class="book-list">
        <!-- Books will be displayed here dynamically -->
    </ul>

    <!-- Pagination -->
    <div id="pagination" class="pagination"></div>

    <script>
        let currentPage = 1;
        const pageSize = 10;

        // Function to update pagination with ellipses and limited page numbers
function updatePagination(totalBooks, currentPage) {
    const pagination = document.getElementById('pagination');
    pagination.innerHTML = ''; // Clear current pagination

    const totalPages = Math.ceil(totalBooks / pageSize);
    const maxVisiblePages = 5; // Maximum number of page buttons to display
```

```
// Previous button
const prevButton = document.createElement('button');
prevButton.innerText = 'Prev';
prevButton.disabled = currentPage === 1;
prevButton.onclick = function() {
  if (currentPage > 1) {
    currentPage--;
    fetchBooks();
  }
};
pagination.appendChild(prevButton);

// Display limited page numbers
let startPage, endPage;
if (totalPages <= maxVisiblePages) {
  // Show all pages if totalPages is less than or equal to maxVisiblePages
  startPage = 1;
  endPage = totalPages;
} else {
  // Otherwise, show a range around the current page
  if (currentPage <= 3) {
    startPage = 1;
    endPage = maxVisiblePages;
  } else if (currentPage + 2 >= totalPages) {
    startPage = totalPages - maxVisiblePages + 1;
    endPage = totalPages;
  } else {
    startPage = currentPage - 2;
    endPage = currentPage + 2;
  }
}
```

```
// Add page buttons
for (let page = startPage; page <= endPage; page++) {
  const button = document.createElement('button');
  button.innerText = page;
  button.onclick = function() {
    currentPage = page;
    fetchBooks();
  };
  if (page === currentPage) {
    button.style.fontWeight = 'bold'; // Highlight current page
  }
  pagination.appendChild(button);
}

// Add ellipses if there are skipped pages
if (startPage > 1) {
  const ellipses = document.createElement('span');
  ellipses.innerText = '...';
  pagination.insertBefore(ellipses, pagination.firstChild);
}
if (endPage < totalPages) {
  const ellipses = document.createElement('span');
  ellipses.innerText = '...';
  pagination.appendChild(ellipses);
}

// Next button
const nextButton = document.createElement('button');
nextButton.innerText = 'Next';
nextButton.disabled = currentPage === totalPages;
nextButton.onclick = function() {
  if (currentPage < totalPages) {
```

```
            currentPage++;
            fetchBooks();
        }
    };
    pagination.appendChild(nextButton);
}

// Function to fetch and display books
function fetchBooks() {
    const searchTerm = document.getElementById('searchTerm').value;
    const sortBy = document.getElementById('sortBy').value;
    const sortOrder = document.getElementById('sortOrder').value;

    // Encode the search term to handle special characters
    const encodedSearchTerm = encodeURIComponent(searchTerm);

    // Prepare the API URL with query parameters
    let apiUrl =
`http://127.0.0.1:5000/books?search=${encodedSearchTerm}&sort_by=${sortBy}&order=${
sortOrder}&page=${currentPage}&page_size=${pageSize}`;

    // Make the API call
    fetch(apiUrl)
        .then(response => response.json())
        .then(data => {
            const bookList = document.getElementById('bookList');
            bookList.innerHTML = ''; // Clear current book list
            bookList.classList.remove('loaded'); // Remove loaded class before updating

            // Handle case where no books are returned
            if (data.books.length === 0) {
                bookList.innerHTML = '<li>No books found for the given filters.</li>';
            } else {
```

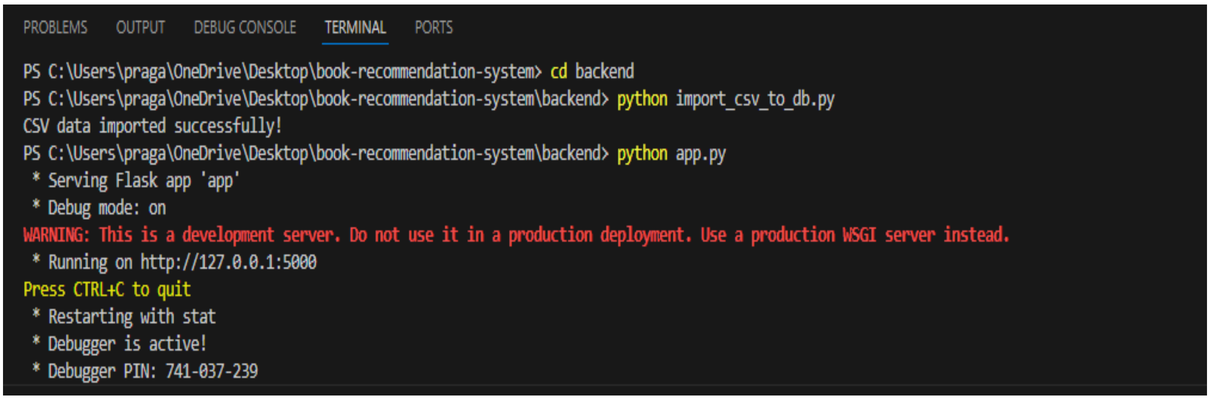
```
        data.books.forEach(book => {
          const bookItem = document.createElement('li');
          bookItem.className = 'book-item';
          bookItem.innerHTML = `
            <h3>${book.title}</h3>
            <p><strong>Author:</strong> ${book.author || 'N/A'}</p>
            <p><strong>Description:</strong> ${book.description || 'No description
available'}</p>
            <p><strong>Genres:</strong> ${book.genres || 'N/A'}</p>
            <p><strong>Avg Rating:</strong> ${book.avg_rating !== null ?
book.avg_rating : 'N/A'}</p>
            <p><strong>Num Ratings:</strong> ${book.num_ratings !== null ?
book.num_ratings : 'N/A'}</p>
            <p><strong>URL:</strong> <a href="${book.url}" target="_blank">View
Book</a></p>
          `;
          bookList.appendChild(bookItem);
        });
        updatePagination(data.totalBooks, currentPage);
      }

      bookList.classList.add('loaded'); // Add loaded class to make it visible
    })
    .catch(error => console.error('Error fetching books:', error));
}

    // Fetch books when the page loads
    window.onload = function() {
      fetchBooks();
    };
  </script>
</body>
</html>
```

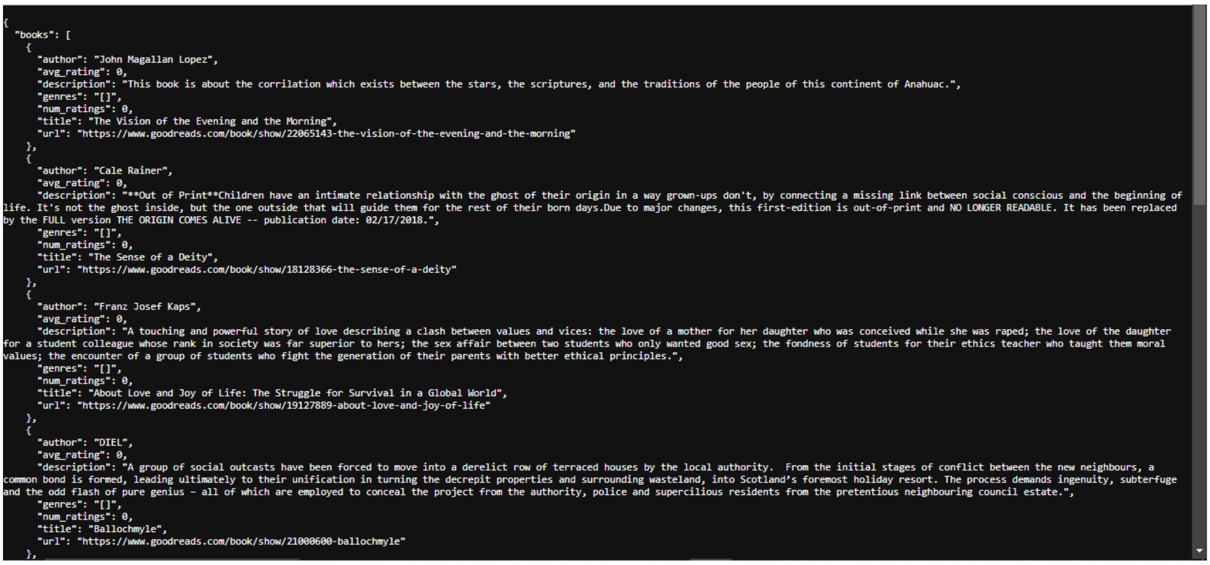
5.PROJECT SCREENSHOTS

IMPORTING DATA AND OPENING WEBSITE



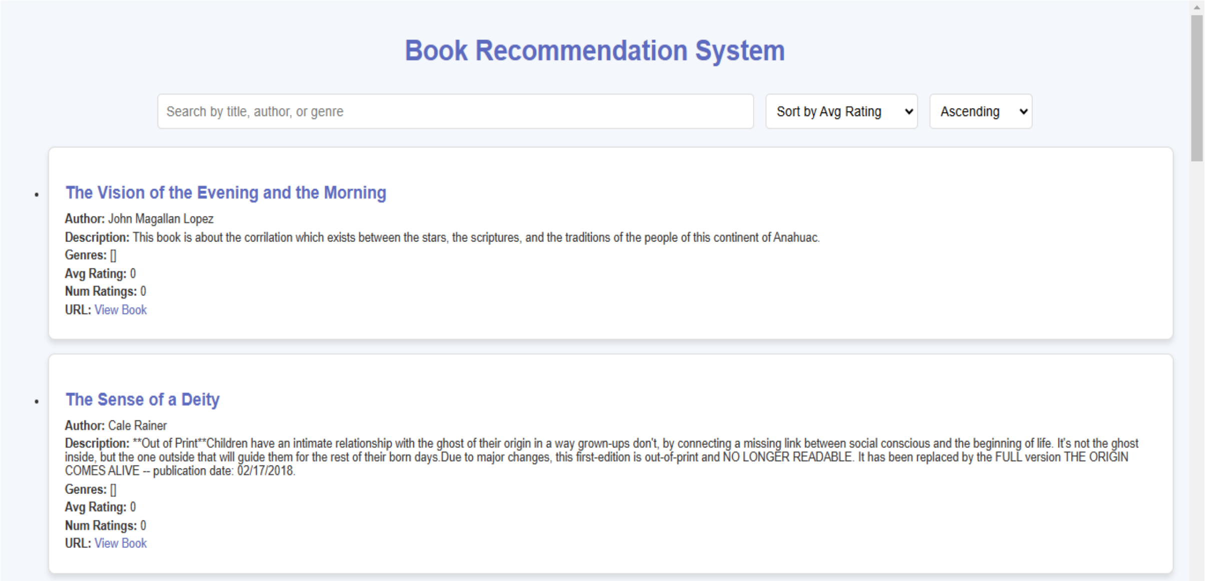
```
PS C:\Users\praga\OneDrive\Desktop\book-recommendation-system> cd backend
PS C:\Users\praga\OneDrive\Desktop\book-recommendation-system\backend> python import_csv_to_db.py
CSV data imported successfully!
PS C:\Users\praga\OneDrive\Desktop\book-recommendation-system\backend> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 741-037-239
```

DATABASE ON BACKEND



```
{
  "books": [
    {
      "author": "John Magallan Lopez",
      "avg_rating": 0,
      "description": "This book is about the corrilation which exists between the stars, the scriptures, and the traditions of the people of this continent of Anahuac.",
      "genres": "[]",
      "num_ratings": 0,
      "title": "The Vision of the Evening and the Morning",
      "url": "https://www.goodreads.com/book/show/22065143-the-vision-of-the-evening-and-the-morning"
    },
    {
      "author": "Cale Rainer",
      "avg_rating": 0,
      "description": "**Out of Print**Children have an intimate relationship with the ghost of their origin in a way grown-ups don't, by connecting a missing link between social conscious and the beginning of
life. It's not the ghost inside, but the one outside that will guide them for the rest of their born days.Due to major changes, this first-edition is out-of-print and NO LONGER READABLE. It has been replaced
by the FULL version THE ORIGIN COMES ALIVE -- publication date: 02/17/2018.",
      "genres": "[]",
      "num_ratings": 0,
      "title": "The Sense of a Deity",
      "url": "https://www.goodreads.com/book/show/18128366-the-sense-of-a-deity"
    },
    {
      "author": "Franz Josef Kaps",
      "avg_rating": 0,
      "description": "A touching and powerful story of love describing a clash between values and vices: the love of a mother for her daughter who was conceived while she was raped; the love of the daughter
for a student colleague whose rank in society was far superior to hers; the sex affair between two students who only wanted good sex; the fondness of students for their ethics teacher who taught them moral
values; the encounter of a group of students who fight the generation of their parents with better ethical principles.",
      "genres": "[]",
      "num_ratings": 0,
      "title": "About Love and Joy of Life: The Struggle for Survival in a Global World",
      "url": "https://www.goodreads.com/book/show/19127889-about-love-and-joy-of-life"
    },
    {
      "author": "DIEL",
      "avg_rating": 0,
      "description": "A group of social outcasts have been forced to move into a derelict row of terraced houses by the local authority.  From the initial stages of conflict between the new neighbours, a
common bond is formed, leading ultimately to their unification in turning the decrepit properties and surrounding wasteland, into Scotland’s foremost holiday resort. The process demands ingenuity, subterfuge
and the odd flash of pure genius – all of which are employed to conceal the project from the authority, police and supercilious residents from the pretentious neighbouring council estate.",
      "genres": "[]",
      "num_ratings": 0,
      "title": "Ballochmyle",
      "url": "https://www.goodreads.com/book/show/21000600-ballochmyle"
    },
```

DEFAULT RECOMMENDING PAGE



Book Recommendation System

- **The Vision of the Evening and the Morning**

  Author: John Magallan Lopez

  Description: This book is about the corrilation which exists between the stars, the scriptures, and the traditions of the people of this continent of Anahuac.

  Genres: []

  Avg Rating: 0

  Num Ratings: 0

  URL: View Book

- **The Sense of a Deity**

  Author: Cale Rainer

  Description: **Out of Print**Children have an intimate relationship with the ghost of their origin in a way grown-ups don't, by connecting a missing link between social conscious and the beginning of life. It's not the ghost inside, but the one outside that will guide them for the rest of their born days.Due to major changes, this first-edition is out-of-print and NO LONGER READABLE. It has been replaced by the FULL version THE ORIGIN COMES ALIVE -- publication date: 02/17/2018.

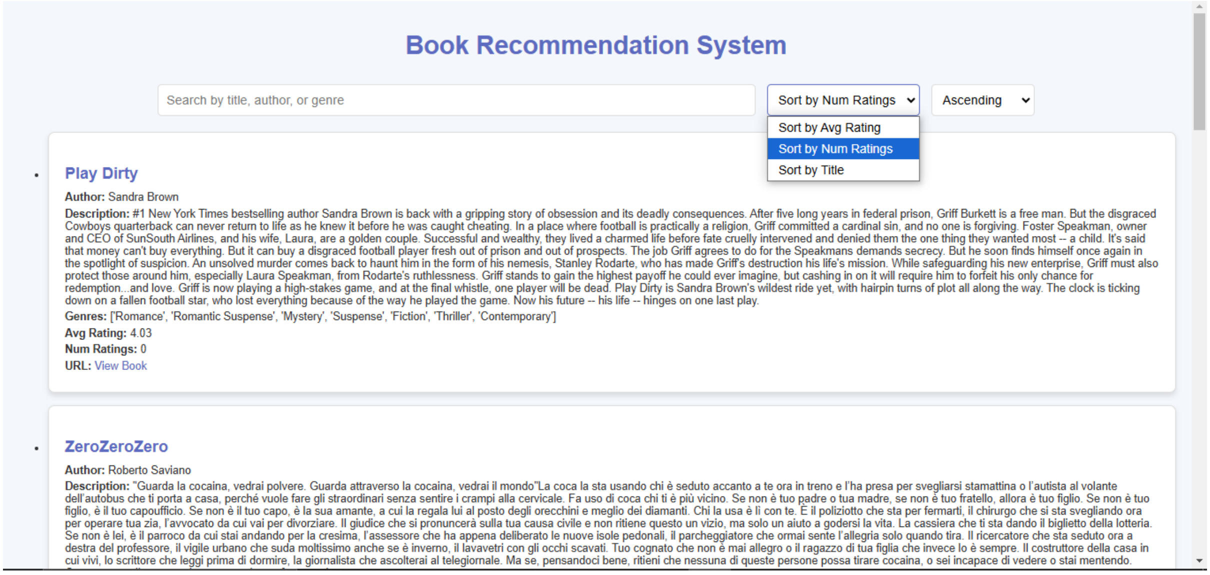
  Genres: []

  Avg Rating: 0

  Num Ratings: 0

  URL: View Book

SORTING BY NUM RATINGS



Book Recommendation System

- **Play Dirty**

  Author: Sandra Brown

  Description: #1 New York Times bestselling author Sandra Brown is back with a gripping story of obsession and its deadly consequences. After five long years in federal prison, Griff Burkett is a free man. But the disgraced Cowboys quarterback can never return to life as he knew it before he was caught cheating. In a place where football is practically a religion, Griff committed a cardinal sin, and no one is forgiving. Foster Speakman, owner and CEO of SunSouth Airlines, and his wife, Laura, are a golden couple. Successful and wealthy, they lived a charmed life before fate cruelly intervened and denied them the one thing they wanted most -- a child. It's said that money can't buy everything. But it can buy a disgraced football player fresh out of prison and out of prospects. The job Griff agrees to do for the Speakmans demands secrecy. But he soon finds himself once again in the spotlight of suspicion. An unsolved murder comes back to haunt him in the form of his nemesis, Stanley Rodarte, who has made Griff's destruction his life's mission. While safeguarding his new enterprise, Griff must also protect those around him, especially Laura Speakman, from Rodarte's ruthlessness. Griff stands to gain the highest payoff he could ever imagine, but cashing in on it will require him to forfeit his only chance for redemption...and love. Griff is now playing a high-stakes game, and at the final whistle, one player will be dead. Play Dirty is Sandra Brown's wildest ride yet, with hairpin turns of plot all along the way. The clock is ticking down on a fallen football star, who lost everything because of the way he played the game. Now his future -- his life -- hinges on one last play.

  Genres: ['Romance', 'Romantic Suspense', 'Mystery', 'Suspense', 'Fiction', 'Thriller', 'Contemporary']

  Avg Rating: 4.03

  Num Ratings: 0

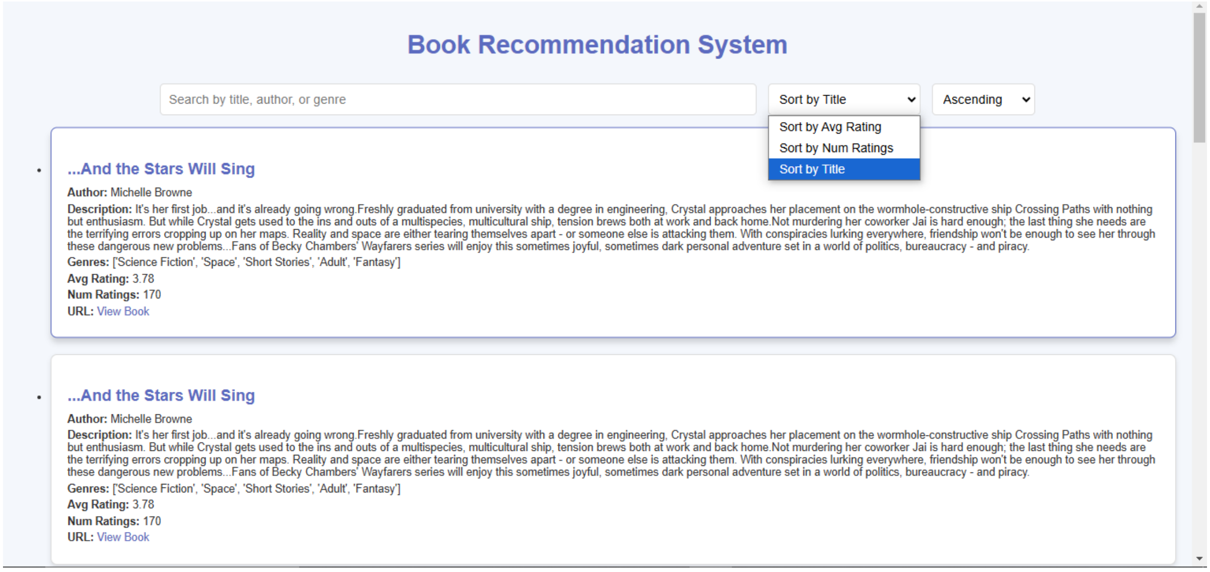
  URL: View Book

- **ZeroZeroZero**

  Author: Roberto Saviano

  Description: "Guarda la cocaina, vedrai polvere. Guarda attraverso la cocaina, vedrai il mondo"La coca la sta usando chi è seduto accanto a te ora in treno e l'ha presa per svegliarsi stamattina o l'autista al volante dell'autobus che ti porta a casa, perché vuole fare gli straordinari senza sentire i crampi alla cervicale. Fa uso di coca chi ti è più vicino. Se non è tuo padre o tua madre, se non è tuo fratello, allora è tuo figlio. Se non è tuo figlio, è il tuo capoufficio. Se non è il tuo capo, è la sua amante, a cui la regala lui al posto degli orecchini e meglio dei diamanti. Chi la usa è lì con te. È il poliziotto che sta per fermarti, il chirurgo che si sta svegliando ora per operare tua zia, l'avvocato da cui vai per divorziare. Il giudice che si pronuncerà sulla tua causa civile e non ritiene questo un vizio, ma solo un aiuto a godersi la vita. La cassiera che ti sta dando il biglietto della lotteria. Se non è lei, è il parroco da cui stai andando per la cresima, l'assessore che ha appena deliberato le nuove isole pedonali, il parcheggiatore che ormai sente l'allegria solo quando tira. Il ricercatore che sta seduto ora a destra del professore, il vigile urbano che suda moltissimo anche se è inverno, il lavavetri con gli occhi scavati. Tuo cognato che non è mai allegro o il ragazzo di tua figlia che invece lo è sempre. Il costruttore della casa in cui vivi, lo scrittore che leggi prima di dormire, la giornalista che ascolterai al telegiornale. Ma se, pensandoci bene, ritieni che nessuna di queste persone possa tirare cocaina, o sei incapace di vedere o stai mentendo.

SORTING BY TITLE



Book Recommendation System

- **...And the Stars Will Sing**

  Author: Michelle Browne

  Description: It's her first job...and it's already going wrong.Freshly graduated from university with a degree in engineering, Crystal approaches her placement on the wormhole-constructive ship Crossing Paths with nothing but enthusiasm. But while Crystal gets used to the ins and outs of a multispecies, multicultural ship, tension brews both at work and back home.Not murdering her coworker Jai is hard enough; the last thing she needs are the terrifying errors cropping up on her maps. Reality and space are either tearing themselves apart - or someone else is attacking them. With conspiracies lurking everywhere, friendship won't be enough to see her through these dangerous new problems...Fans of Becky Chambers' Wayfarers series will enjoy this sometimes joyful, sometimes dark personal adventure set in a world of politics, bureaucracy - and piracy.

  Genres: ['Science Fiction', 'Space', 'Short Stories', 'Adult', 'Fantasy']

  Avg Rating: 3.78

  Num Ratings: 170

  URL: View Book

- **...And the Stars Will Sing**

  Author: Michelle Browne

  Description: It's her first job...and it's already going wrong.Freshly graduated from university with a degree in engineering, Crystal approaches her placement on the wormhole-constructive ship Crossing Paths with nothing but enthusiasm. But while Crystal gets used to the ins and outs of a multispecies, multicultural ship, tension brews both at work and back home.Not murdering her coworker Jai is hard enough; the last thing she needs are the terrifying errors cropping up on her maps. Reality and space are either tearing themselves apart - or someone else is attacking them. With conspiracies lurking everywhere, friendship won't be enough to see her through these dangerous new problems...Fans of Becky Chambers' Wayfarers series will enjoy this sometimes joyful, sometimes dark personal adventure set in a world of politics, bureaucracy - and piracy.

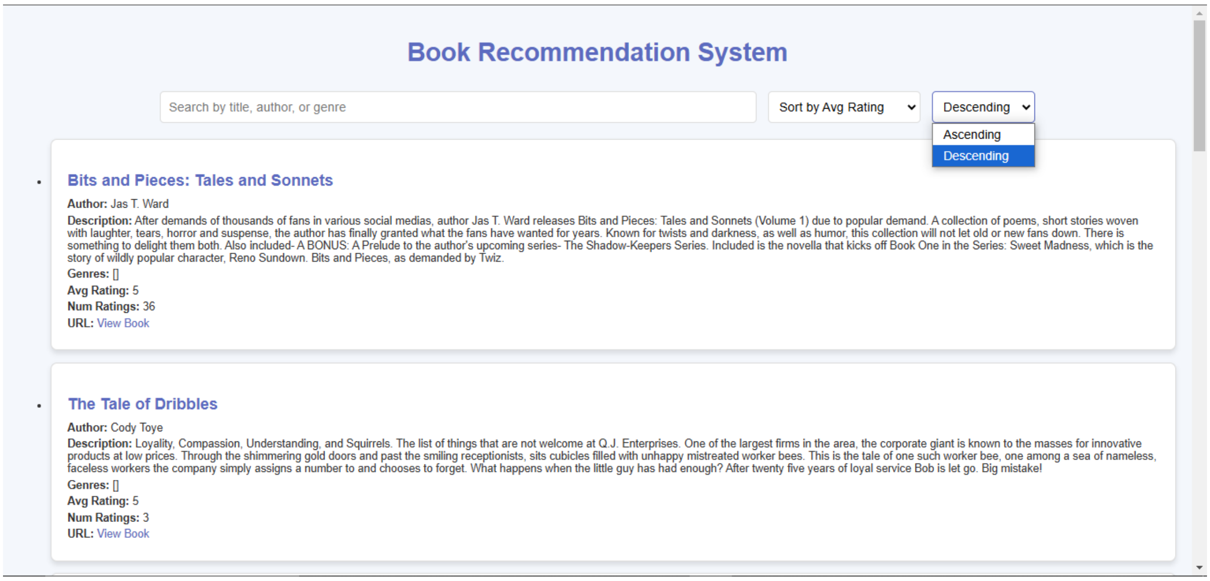
  Genres: ['Science Fiction', 'Space', 'Short Stories', 'Adult', 'Fantasy']

  Avg Rating: 3.78

  Num Ratings: 170

  URL: View Book

BY DESCENDING ORDER



Book Recommendation System

- **Bits and Pieces: Tales and Sonnets**

  Author: Jas T. Ward

  Description: After demands of thousands of fans in various social medias, author Jas T. Ward releases Bits and Pieces: Tales and Sonnets (Volume 1) due to popular demand. A collection of poems, short stories woven with laughter, tears, horror and suspense, the author has finally granted what the fans have wanted for years. Known for twists and darkness, as well as humor, this collection will not let old or new fans down. There is something to delight them both. Also included- A BONUS: A Prelude to the author's upcoming series- The Shadow-Keepers Series. Included is the novella that kicks off Book One in the Series: Sweet Madness, which is the story of wildly popular character, Reno Sundown. Bits and Pieces, as demanded by Twiz.

  Genres: []

  Avg Rating: 5

  Num Ratings: 36

  URL: View Book

- **The Tale of Dribbles**

  Author: Cody Toye

  Description: Loyality, Compassion, Understanding, and Squirrels. The list of things that are not welcome at Q.J. Enterprises. One of the largest firms in the area, the corporate giant is known to the masses for innovative products at low prices. Through the shimmering gold doors and past the smiling receptionists, sits cubicles filled with unhappy mistreated worker bees. This is the tale of one such worker bee, one among a sea of nameless, faceless workers the company simply assigns a number to and chooses to forget. What happens when the little guy has had enough? After twenty five years of loyal service Bob is let go. Big mistake!

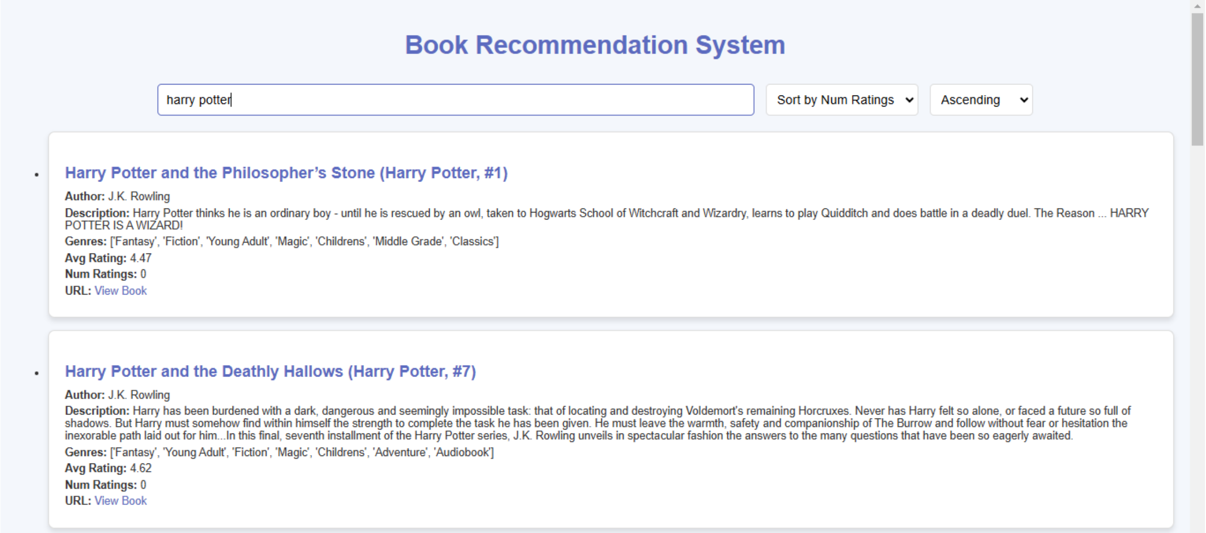
  Genres: []

  Avg Rating: 5

  Num Ratings: 3

  URL: View Book

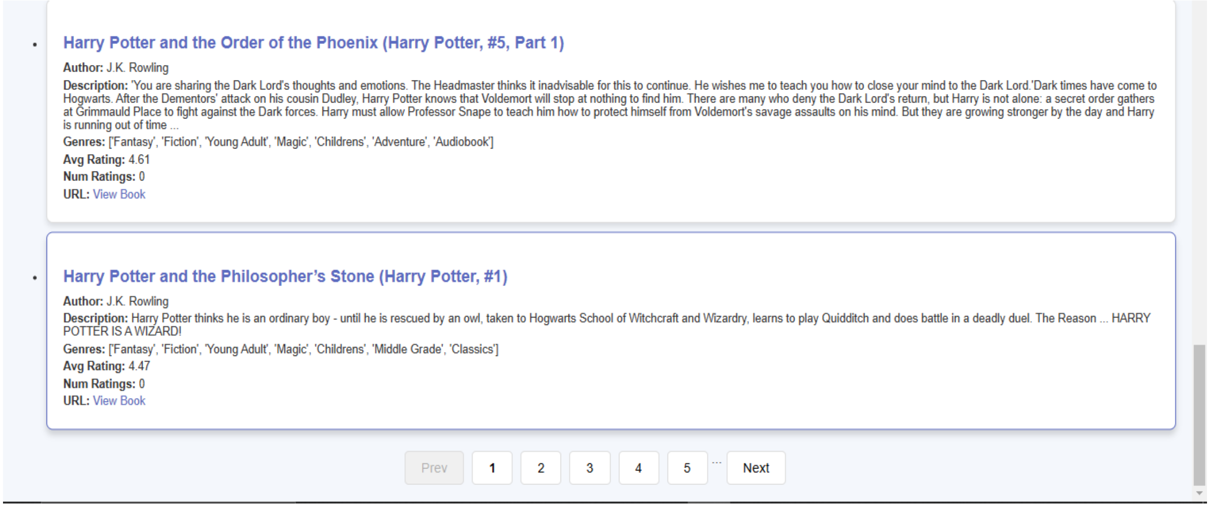
SEARCHING



- **Harry Potter and the Philosopher’s Stone (Harry Potter, #1)**
  Author: J.K. Rowling
  Description: Harry Potter thinks he is an ordinary boy - until he is rescued by an owl, taken to Hogwarts School of Witchcraft and Wizardry, learns to play Quidditch and does battle in a deadly duel. The Reason ... HARRY POTTER IS A WIZARD!
  Genres: ['Fantasy', 'Fiction', 'Young Adult', 'Magic', 'Childrens', 'Middle Grade', 'Classics']
  Avg Rating: 4.47
  Num Ratings: 0
  URL: View Book

- **Harry Potter and the Deathly Hallows (Harry Potter, #7)**
  Author: J.K. Rowling
  Description: Harry has been burdened with a dark, dangerous and seemingly impossible task: that of locating and destroying Voldemort's remaining Horcruxes. Never has Harry felt so alone, or faced a future so full of shadows. But Harry must somehow find within himself the strength to complete the task he has been given. He must leave the warmth, safety and companionship of The Burrow and follow without fear or hesitation the inexorable path laid out for him...In this final, seventh installment of the Harry Potter series, J.K. Rowling unveils in spectacular fashion the answers to the many questions that have been so eagerly awaited.
  Genres: ['Fantasy', 'Young Adult', 'Fiction', 'Magic', 'Childrens', 'Adventure', 'Audiobook']
  Avg Rating: 4.62
  Num Ratings: 0
  URL: View Book

PAGE FUNCTIONALITY



- **Harry Potter and the Order of the Phoenix (Harry Potter, #5, Part 1)**
  Author: J.K. Rowling
  Description: 'You are sharing the Dark Lord's thoughts and emotions. The Headmaster thinks it inadvisable for this to continue. He wishes me to teach you how to close your mind to the Dark Lord.'Dark times have come to Hogwarts. After the Dementors' attack on his cousin Dudley, Harry Potter knows that Voldemort will stop at nothing to find him. There are many who deny the Dark Lord's return, but Harry is not alone: a secret order gathers at Grimmauld Place to fight against the Dark forces. Harry must allow Professor Snape to teach him how to protect himself from Voldemort's savage assaults on his mind. But they are growing stronger by the day and Harry is running out of time ...
  Genres: ['Fantasy', 'Fiction', 'Young Adult', 'Magic', 'Childrens', 'Adventure', 'Audiobook']
  Avg Rating: 4.61
  Num Ratings: 0
  URL: View Book

- **Harry Potter and the Philosopher’s Stone (Harry Potter, #1)**
  Author: J.K. Rowling
  Description: Harry Potter thinks he is an ordinary boy - until he is rescued by an owl, taken to Hogwarts School of Witchcraft and Wizardry, learns to play Quidditch and does battle in a deadly duel. The Reason ... HARRY POTTER IS A WIZARD!
  Genres: ['Fantasy', 'Fiction', 'Young Adult', 'Magic', 'Childrens', 'Middle Grade', 'Classics']
  Avg Rating: 4.47
  Num Ratings: 0
  URL: View Book

6. CONCLUSION

The **Book Recommendation System** project successfully achieves its primary goal of providing a user-friendly platform for users to search, sort, and explore a variety of books. By leveraging a Flask backend and a MySQL database, the system efficiently handles user requests, ensuring quick responses and an intuitive browsing experience.

This project highlights the following achievements and insights:

- **Effective Data Management**: The application efficiently manages book data, offering users the ability to search and filter results based on their interests.
- **Scalable Design**: Built with a modular code structure, the application can be easily expanded with additional features like more advanced recommendation algorithms or user profile customization.
- **Performance Optimization**: The backend queries and frontend interaction are optimized for fast loading and smooth user interactions, making the application accessible even with extensive datasets.

Future Prospects

To enhance the application further, future improvements could focus on:

- **Advanced Recommendation Algorithms**: Incorporating machine learning algorithms for personalized book recommendations.
- **UI/UX Improvements**: Adding visual elements like book covers, user reviews, and interactive ratings could enrich the user experience.
- **Enhanced Security**: Implementing user authentication and data encryption can make the application more secure, especially for personalizing recommendations.

In conclusion, the **Book Recommendation System** provides a solid foundation for an engaging and scalable book discovery platform, serving as a valuable resource for avid readers and casual users alike. The project's success demonstrates effective integration of database management, backend functionality, and a responsive user interface, paving the way for future development and enhanced features.

7. REFERENCES

1. **Flask Documentation**: Flask provides the main framework for routing and backend management.
   - Flask Documentation. (n.d.). Retrieved from https://flask.palletsprojects.com/
2. **MySQL Documentation**: MySQL is used as the backend database, managing book data storage and retrieval.
   - MySQL Documentation. (n.d.). Retrieved from https://dev.mysql.com/doc/
3. **JavaScript Fetch API**: The Fetch API is used for asynchronous requests from the frontend to the backend.
   - MDN Web Docs. Fetch API. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
4. **HTML and CSS Basics**: HTML and CSS form the foundation of the application's frontend, creating a responsive layout for users.
   - MDN Web Docs. HTML Basics. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/HTML
   - MDN Web Docs. CSS Basics. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/CSS
5. **CSV Module (Python)**: The CSV module in Python is used for importing data from CSV files into the database.
   - Python Documentation. CSV Module. Retrieved from https://docs.python.org/3/library/csv.html
6. **SQL Syntax and Commands**: The SQL commands in schema.sql set up the database schema and define the necessary tables.
   - W3Schools. SQL Tutorial. Retrieved from https://www.w3schools.com/sql/