

PROJECT REPORT ON LIBRARY MANAGEMENT SYSTEM

GROUP MEMBERS

Kalai Arasi Jayakumar	801388846
Rajeswari Jayachandran	801376535
Amrutha Reddy Karumuru	801387628
Sai Harsha Chapala	801391020
Sai Adhithi Kesari	801393147

INDEX

CONTENT	PAGE NO
ABSTRACT	2
INTRODUCTION	3
SOFTWARE REQUIREMENTS	5
LITERATURE REVIEW	8
METHODOLOGY	10
CODE DESCRIPTION	15
RESULTS	16
PROBLEM ANALYSIS	22
FUTURE SCOPE	23
CONCLUSION	24

ABSTRACT

The library management system presented in this project is a sophisticated and user-centric platform developed using Python's Flask framework for the backend and Streamlit for the frontend. It offers a seamless and intuitive interface for administrators and users alike, aiming to revolutionize the way libraries function and interact with their patrons. The system's foundation lies in robust user authentication mechanisms, ensuring secure access and personalized experiences upon login.

Key features of the system include the Google Book Finder module, which leverages the Global Search Google API to provide users with powerful search capabilities. This module enables users to conduct comprehensive searches based on book titles or author names, enhancing the efficiency and accuracy of book discovery within the library's collection. Additionally, the system's Catalog module serves as a centralized hub for managing the library's book inventory. It provides users with real-time access to an up-to-date catalog of available books, streamlining the browsing, selection, and borrowing processes.

The Analysis module complements these functionalities by harnessing data analytics techniques to generate insightful graphical representations of book-related data. Users can gain valuable insights into trends, ratings, usage patterns, and popular genres, empowering them to make informed decisions and tailor their library experience to their preferences. Furthermore, the Book Manager module introduces efficient workflows for book borrowing and returning, simplifying the process for users while enabling administrators to effectively manage inventory and monitor borrowing activities.

Lastly, the Feedback module fosters user engagement by providing a platform for users to share feedback, ratings, and suggestions. This interactive feedback loop not only enhances user satisfaction but also serves as a valuable tool for continuous improvement, helping libraries adapt to the evolving needs and preferences of their patrons. Overall, the library management system exemplifies a fusion of innovative technologies, user-centric design principles, and data-driven insights, aiming to redefine the library experience and elevate the standards of library services.

INTRODUCTION

In today's digital age, libraries play a crucial role in providing access to knowledge and information. To meet the evolving needs of library patrons and administrators, efficient and user-friendly library management systems are essential. The library management system presented in this project is a comprehensive software solution designed to streamline operations, enhance user experience, and leverage data-driven insights for informed decision-making.

Overview:

The library management system is built upon a solid foundation of modern technologies, including Python with Flask for the backend and Streamlit for the frontend. This combination ensures a robust and scalable system that caters to the diverse needs of libraries of all sizes. The system offers a range of functionalities, from book searching and catalog management to data analysis and user feedback, making it a versatile and indispensable tool for library administrators and patrons alike.

Advantages to End Users:

The system's user-centric design is one of its primary advantages, offering a seamless and intuitive experience for end users. Patrons benefit from efficient book searches using the Google Book Finder module, which integrates the Global Search Google API for comprehensive search capabilities based on book titles or author names. Real-time access to an updated catalog of books ensures users can easily find and explore the library's collection. The Analysis module provides graphical insights into book-related data, empowering users to make informed decisions and discover new books based on trends and ratings. Additionally, streamlined book borrowing and returning processes through the Book Manager module enhance user convenience and satisfaction. The Feedback module fosters user engagement by providing a platform for feedback and ratings, facilitating continuous improvement of library services.

Objectives:

The primary objectives of this project are to develop a robust and user-friendly library management system that integrates backend and frontend technologies seamlessly. The system aims to enhance user experience through intuitive interfaces and efficient workflows, leverage data analytics for informed decision-making and improved library services, and foster user engagement through interactive feedback mechanisms. By achieving these objectives, the system aims to elevate the standards of library services and adapt to the evolving needs of modern libraries and their patrons.

Need for the Project:

The need for this project stems from the challenges faced by traditional library management systems in terms of accessibility, data analysis, and user engagement. With libraries embracing digital transformation, there is a growing demand for modern solutions that optimize operations, provide actionable insights, and enhance user satisfaction. This project addresses these needs by leveraging technology to create a comprehensive library management system that meets the diverse requirements of libraries in the digital age.

Statement of the goal:

The project's objective is to create a productive book lending system that is supplemented with a potent search engine and insights driven by machine learning to recognize patterns like the best-selling books. In addition to managing loans and returns, updating databases with new titles, and giving information on library staff and checked out books, the proposed system will benefit both users and administrators.

SOFTWARE REQUIREMENTS

Backend Development:

Use Python as the programming language.

Utilize Flask as the web framework for backend server development.

Choose either MySQL or PostgreSQL as the database management system for storing book data, user details, and transactions.

Frontend Development:

Employ Streamlit (Version 1.x) for creating the user interface.

Utilize HTML, CSS, and JavaScript for frontend design and functionality.

Data Analysis and Visualization:

Use Pandas (Version 1.x) for data manipulation and analysis.

Employ Matplotlib (Version 3.x) and Seaborn (Version 0.x) for generating graphs, charts, and reports.

Security and Encryption:

Implement Secure Socket Layer (SSL) or Transport Layer Security (TLS) for secure communication (HTTPS).

Use hashlib for hashing passwords and sensitive data.

APIs and Libraries:

Develop a RESTful API for communication between the frontend and backend.

Integrate Google Books API for book search functionality.

Development Environment:

Utilize an Integrated Development Environment (IDE) such as PyCharm or Visual Studio Code.

Use Git for version control and collaboration.

LIBRARIES

Streamlit: Employed as a web application framework to craft interactive user interfaces.

Pandas: Utilized for data manipulation and analysis tasks involving structured data.

Matplotlib: Employed for generating visualizations like graphs and charts.

Seaborn: Utilized for enhanced statistical data visualizations, built upon Matplotlib.

Requests: Utilized for making HTTP requests to external APIs, such as the Google Books API.

Flask: Employed as a micro web framework for backend server logic development and APIs.

Datetime: The Python module designed for handling dates and times, facilitating tasks such as creation, formatting, and manipulation of date and time objects.

Sqlite: An embedded SQL database engine integrated within Python, allowing seamless interaction with SQLite databases without the need for external installations.

Pathlib: A Python module offering an object-oriented interface for managing file system paths, streamlining path manipulation with a more intuitive approach compared to conventional string-based methods.

PREREQUISITIES USED

Tools and Environment:

Programming Languages: Python for backend development, data processing, and scripting.

Web Frameworks: Flask for backend web application development.

Web Development: Streamlit for frontend web application development.

Database Management System: MySQL

Tables names:

Authors:

Author_id

Name

Books:

Book_id

Bookname

Publisher

Rating

Author_id

Feedback:

Review_id

Review

Rating

Book_id

User_id

Loans:

Loan_id

Borrowed_date

Returned

Book_id

User_id

Users:

Id

Username

Password

LITERATURE REVIEW

In recent years, the development of library management systems (LMS) has seen significant advancements, particularly in terms of technology integration, user experience enhancement, and data-driven decision-making. This literature review explores the modern approaches and technologies employed in LMS development, focusing on Python, Flask, Streamlit, Google Book Finder API, data analysis, and user feedback mechanisms.

Python in Backend Development:

Python has emerged as a preferred choice for backend development in various domains due to its simplicity, versatility, and extensive libraries. In the context of LMS, Python offers robust capabilities for data processing, database management, and API development. Libraries such as Flask provide a lightweight yet powerful framework for building RESTful APIs, enabling seamless communication between frontend and backend components of the system.

Streamlit for Frontend Interface:

Streamlit has gained popularity as a user-friendly framework for creating interactive and visually appealing frontend interfaces. Its simplicity in code structure and real-time updates make it ideal for applications like LMS, where users require intuitive navigation, data visualization, and input forms. Streamlit's integration with Python allows developers to create dynamic dashboards and UI elements without extensive frontend development expertise.

Google Book Finder API Integration:

The integration of external APIs, such as the Google Book Finder API, adds significant value to LMS by enhancing search functionalities. Users can leverage global search capabilities to find books based on titles, authors, genres, and other criteria, improving accessibility and user satisfaction. The API's vast database and search algorithms ensure comprehensive and accurate search results, enriching the overall user experience.

Data Analysis and Visualization:

Modern LMS emphasizes data-driven insights and decision-making processes. Technologies like Pandas, Seaborn, and Matplotlib enable developers to perform in-depth data analysis, generate meaningful reports, and visualize trends and patterns. Through interactive graphs, charts, and statistical analysis, administrators and users can gain valuable insights into library usage, popular books, borrowing trends, and more.

User Feedback Mechanisms:

User feedback plays a crucial role in continuous improvement and user satisfaction. LMS platforms integrate feedback forms, rating systems, and user comments to gather insights, identify areas for enhancement, and address user needs effectively. Feedback mechanisms not only engage users but also foster a collaborative environment for system improvement and user-centric development.

In conclusion, the literature review highlights the convergence of Python, Flask, Streamlit, external APIs like Google Book Finder, data analysis tools, and user feedback mechanisms in modern LMS development. These technologies collectively contribute to creating robust, user-friendly, and data-driven library management systems that meet the evolving needs of libraries, administrators, and users in the digital age.

METHODOLOGY

Initiation Phase:

Define project objectives, scope, and requirements based on stakeholder input and analysis.

Identify key stakeholders including users, administrators, developers, and external API providers.

Create Use Case Diagrams to illustrate user interactions and system functionalities.

Develop a detailed project plan outlining timelines, milestones, resources, and deliverables for each phase.

Design Phase:

Database Design:

Utilize a relational database management system like MySQL to store comprehensive book-related data, including titles, authors, publishers, availability status, borrower history, and ratings. Create normalized database tables for books, users, transactions, feedback, and administrative tasks to ensure data integrity and optimize querying efficiency.

User Interface Design:

Design an intuitive and user-friendly interface catering to both administrators and users. Include essential features such as search functionality (by title, author, ISBN), detailed book information display, borrowing and return options, user feedback submission, and administrative task management tools.

System Architecture:

Adopt a client-server architecture with Streamlit as the client interface interacting seamlessly with a Python Flask backend. Implement RESTful APIs for smooth communication between the frontend and backend systems, facilitating efficient data exchange and execution of system functionalities.

Backend Development:

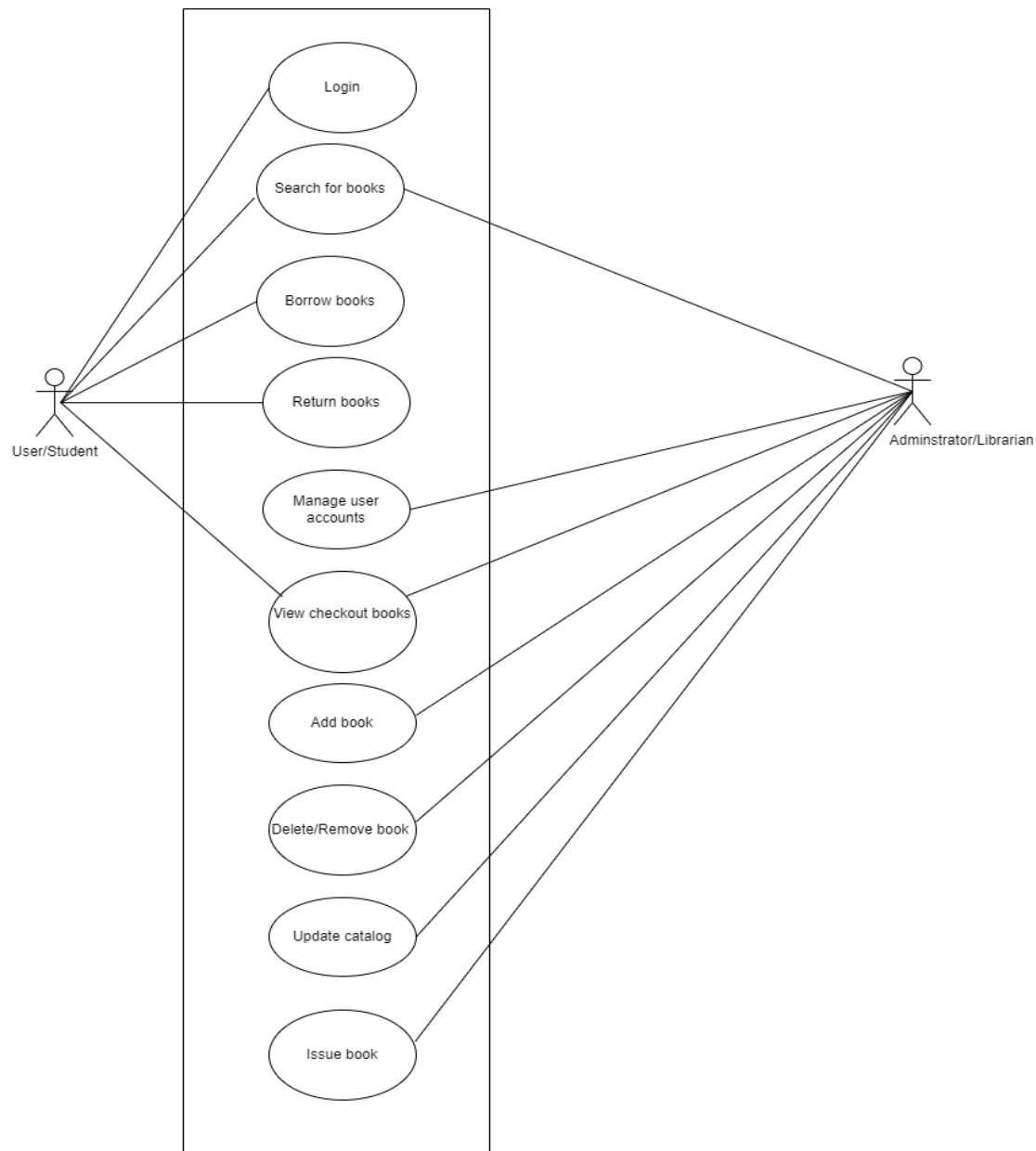
Develop the backend using Python along with the Flask framework to manage core business logic, data processing tasks, and database interactions. Implement modules for book management (e.g., adding, updating, deleting books), user authentication,

transaction handling (borrowing, returning books), and data analysis (e.g., generating reports, analyzing trends).

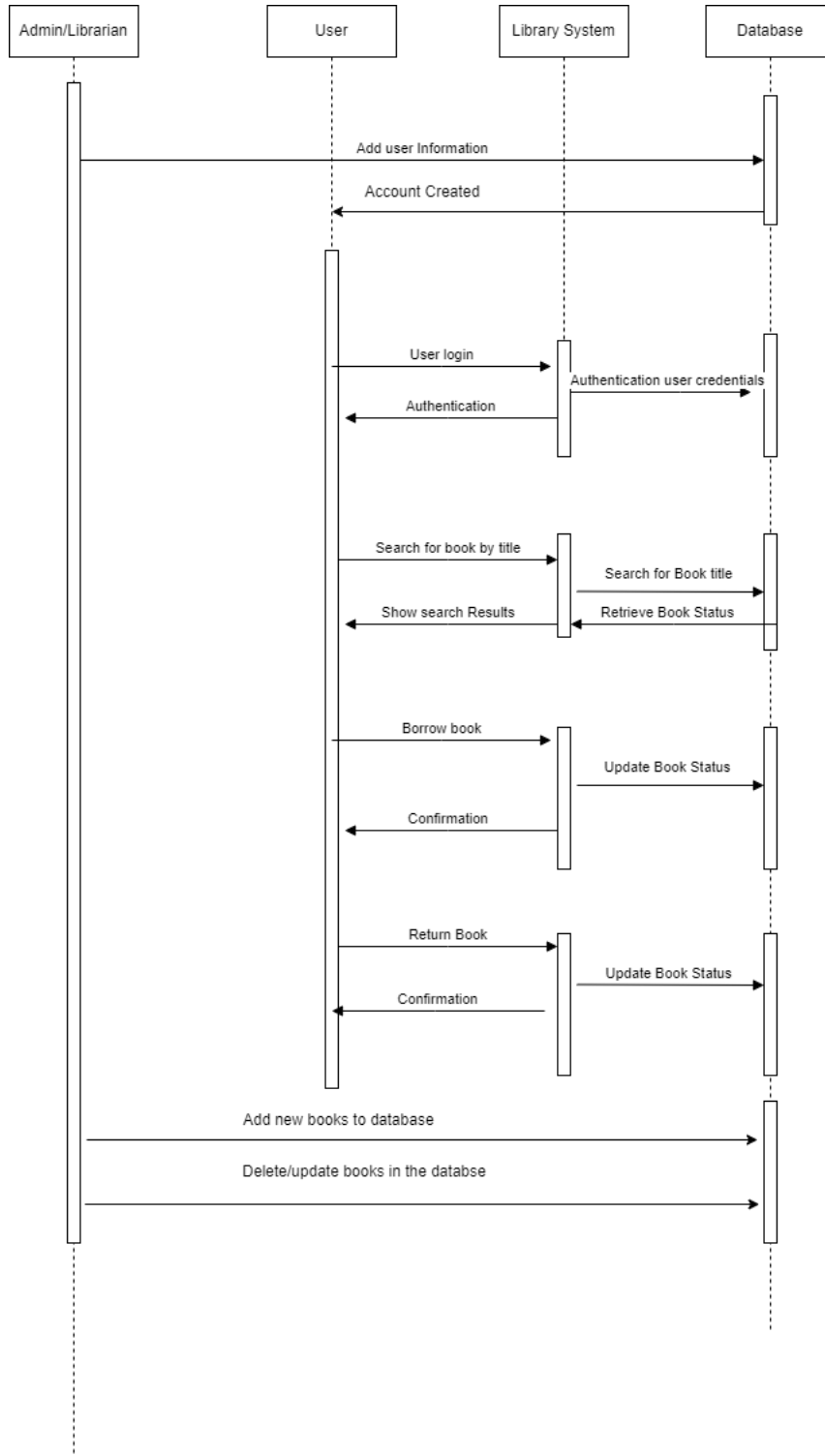
Data Analysis and Reporting:

Integrate data analysis libraries such as Pandas, Matplotlib, and Seaborn into the backend for generating analytical reports and visualizations. Develop algorithms for analyzing book popularity, user preferences, borrowing patterns, and generating actionable insights to support decision-making processes.

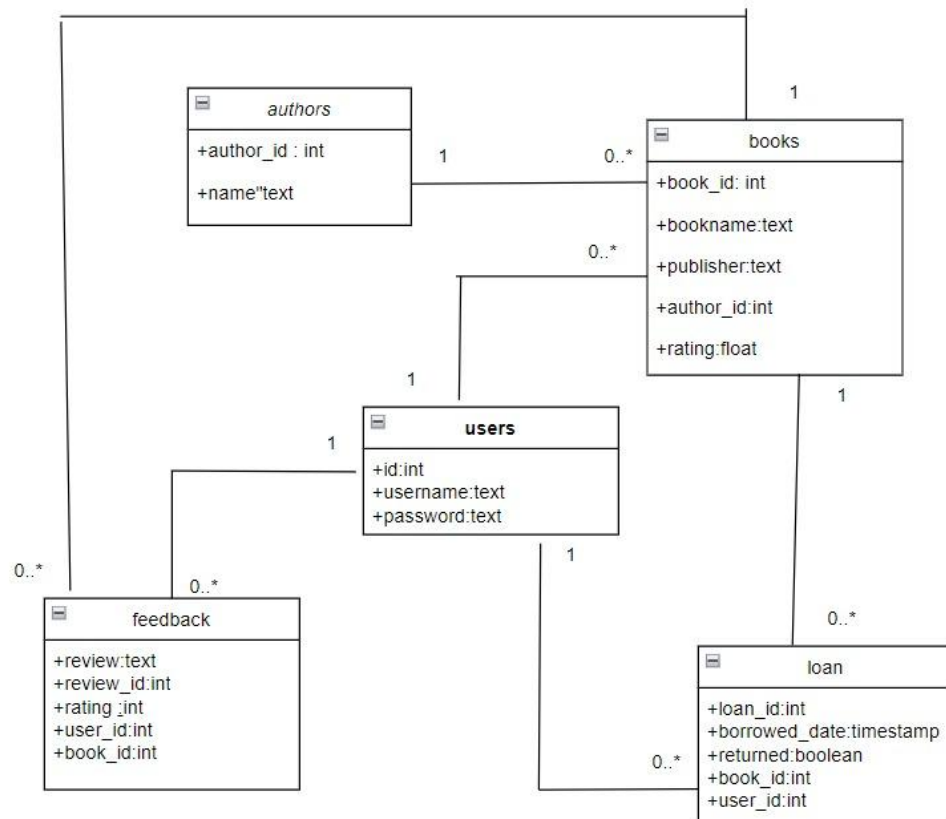
Use Case Diagram:



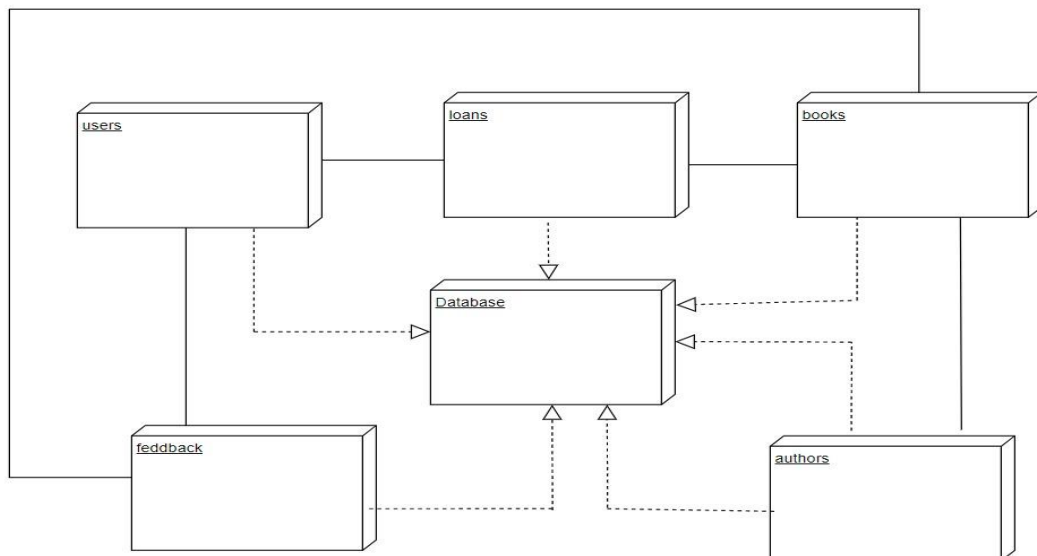
Sequence Diagram:



Class Diagram:



Deployment Diagram:



Development Phase:

Set up Flask backend: Create routes and endpoints for user authentication, book search, catalog management, data analysis, book borrowing/returning, and feedback submission.

Implement front end with Streamlit: Develop the front end using Streamlit to create the user interface based on the UI designs.

Integrate Google Books API: Implement API calls to the Google Books API for book search functionality, handling responses and displaying relevant book details.

Develop database schema based on the UML Class Diagram, create models for entities like User, Book, Borrowing, Feedback, and establish database connections for CRUD operations.

Testing Phase:

Conduct unit testing: Test individual components and functionalities of the system, including authentication, API integration, data storage, and user interactions.

Perform integration testing: Validate interactions between different modules, check data flow, and ensure seamless communication between front end and back end components.

User acceptance testing (UAT): Involve stakeholders and users to test the system's functionalities, gather feedback, and make necessary adjustments based on user input.

Deployment Phase:

Prepare for deployment: Configure the application for deployment on a web server or hosting platform, ensuring compatibility and performance optimization.

Set up security measures: Implement HTTPS, authentication mechanisms, and data encryption to secure user data and transactions.

Conduct final testing: Test the deployed application in the production environment to ensure functionality, security, and user experience.

CODE DESCRIPTION

Imports:

The code starts by importing necessary libraries such as Streamlit for web app creation, Pandas for data handling, Requests for API requests, and Seaborn/Matplotlib for data visualization. Additionally, a custom module for creating a navigation menu is imported.

Sidebar Navigation:

Within a sidebar, a navigation menu is created using a custom function. This menu includes options like Login, Book Finder, Analysis, BookManager, BookNavigation, and Feedback, each with icons and customized styles.

User Login and Admin Login:

The code segment manages user authentication by offering separate login forms for users and administrators. It allows users to select their credentials (User or Admin) and handles authentication accordingly.

Book Dataset Analysis:

This section includes code for analyzing a book dataset, generating various analyses such as top-rated books, most occurring books, top authors, top-rated books by reviews, and top book publishers. The analysis results are visualized using Seaborn and Matplotlib.

Book Manager:

The code related to book management functionalities is provided here, including book checkout, return date generation, data storage, and feedback on successful data storage.

Book Finder:

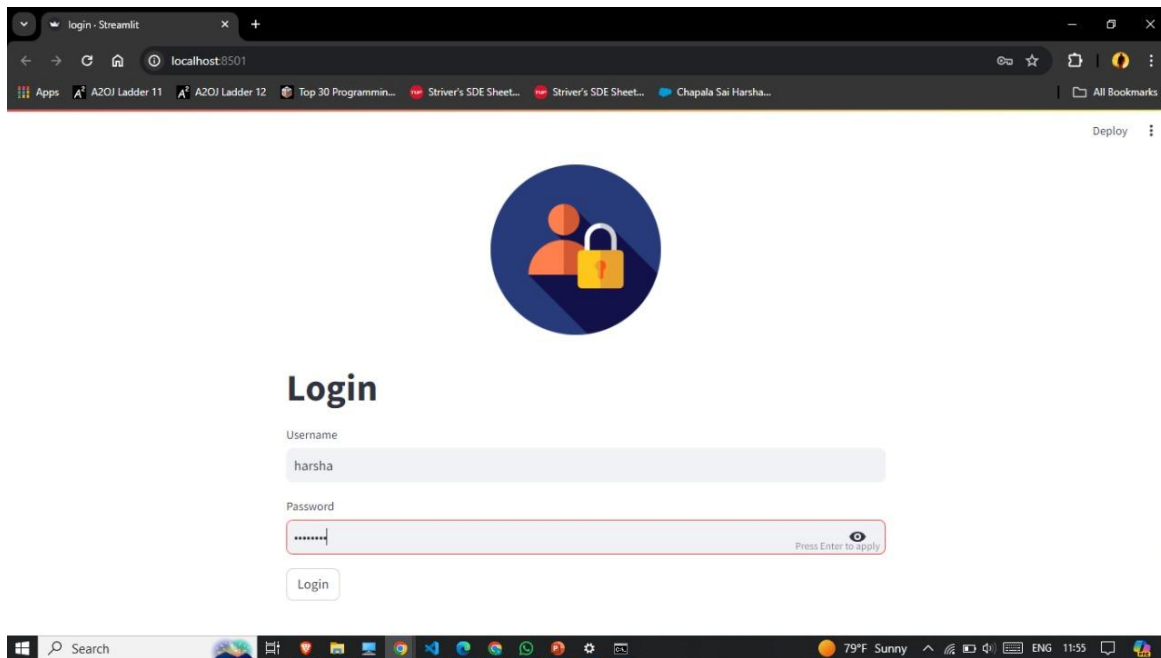
This part of the code deals with book search functionality using the Google Books API. It allows users to search for books by entering book names or author names, displaying search results with book details and descriptions.

Feedback Submission:

Lastly, the code includes functionality for capturing user feedback through a form and storing it in a CSV file for future reference.

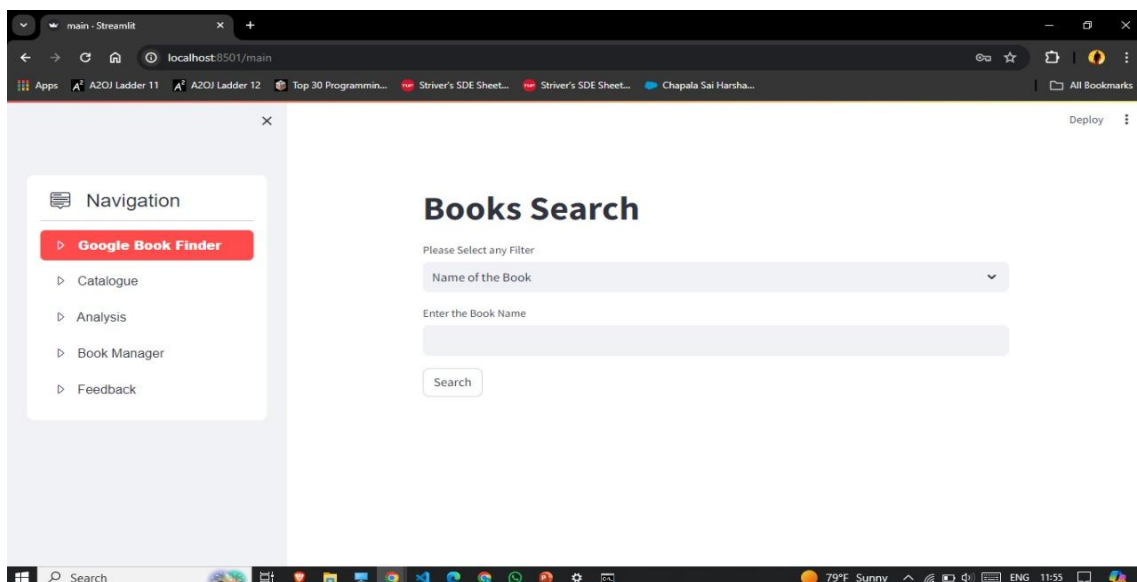
RESULTS

1.LOGIN



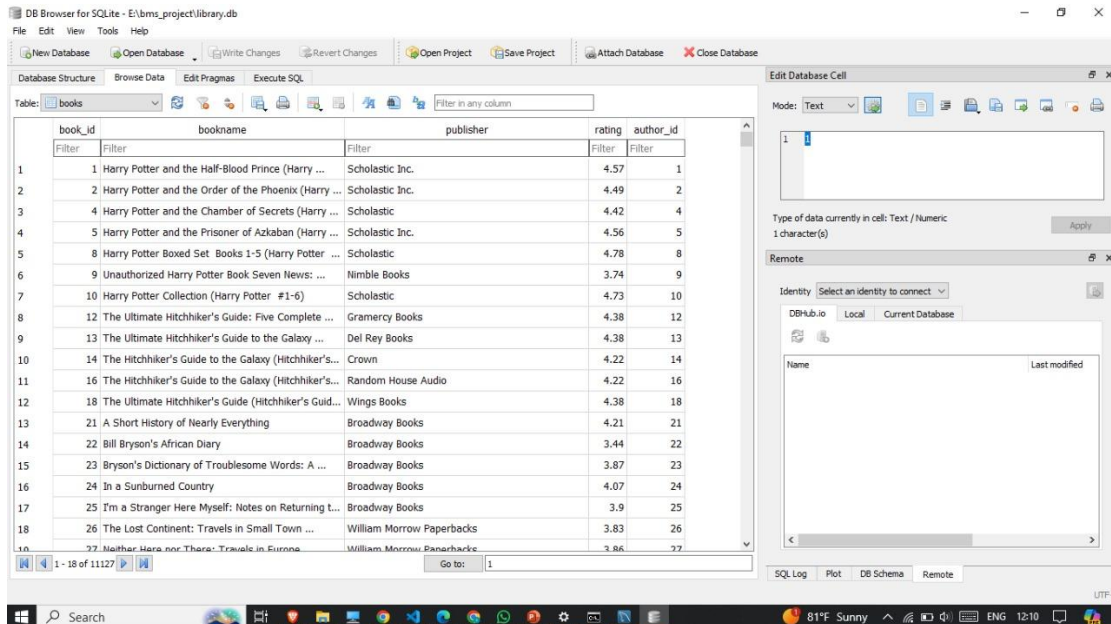
The login page in the code prompts users to enter their username and password. Upon clicking "Login," the system checks these credentials for validity. If correct, users are granted access to the library management system. Otherwise, an error message is displayed. This page serves as the entry point, ensuring secure access to system features based on user roles.

2.GOOGLE BOOK FINDER

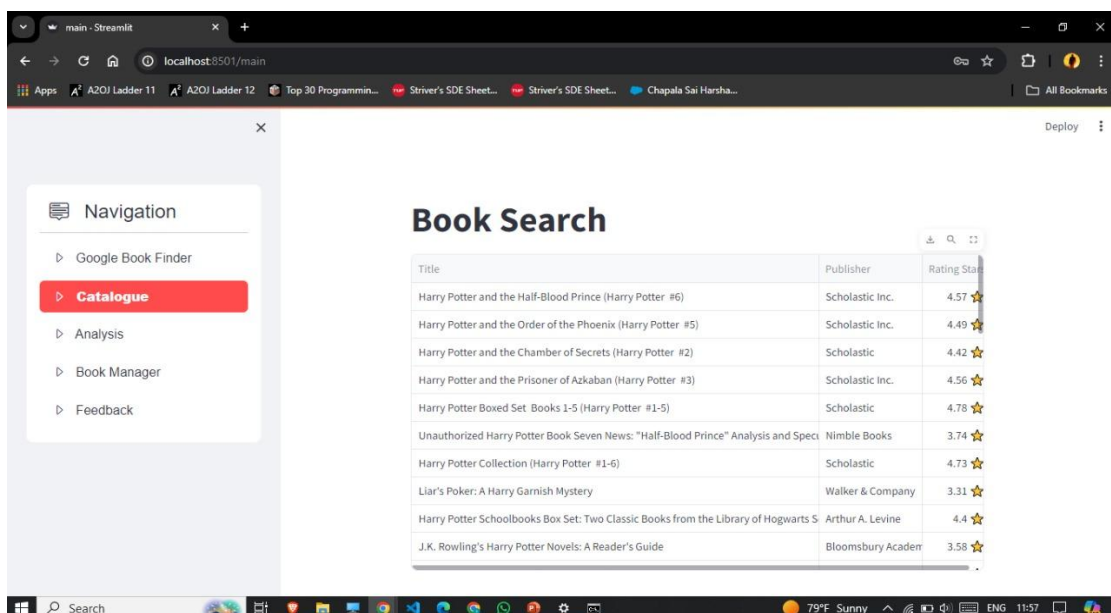


The Google Book Finder feature in the library management system allows users to search for books using the Google Books API. Users can enter either the book's title or the author's name to find relevant books. The system sends a request to the Google Books API, retrieves matching book details such as title, author, description, and thumbnail image, and displays the search results in the application interface. This functionality helps users easily discover and explore books available in the Google Books database.

3.CATALOG

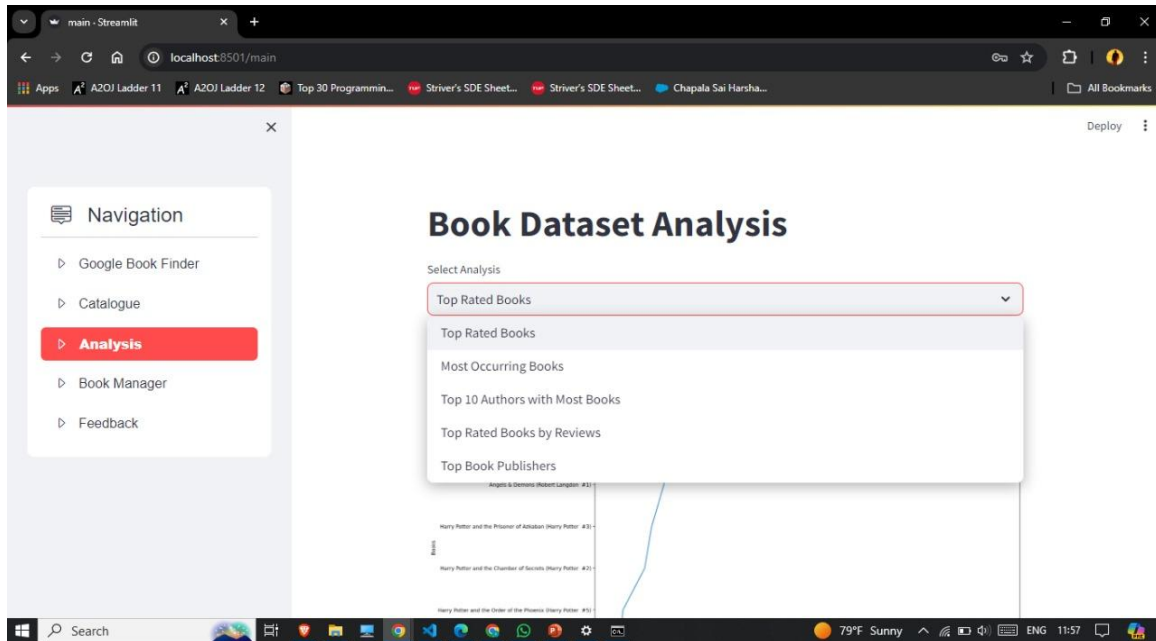


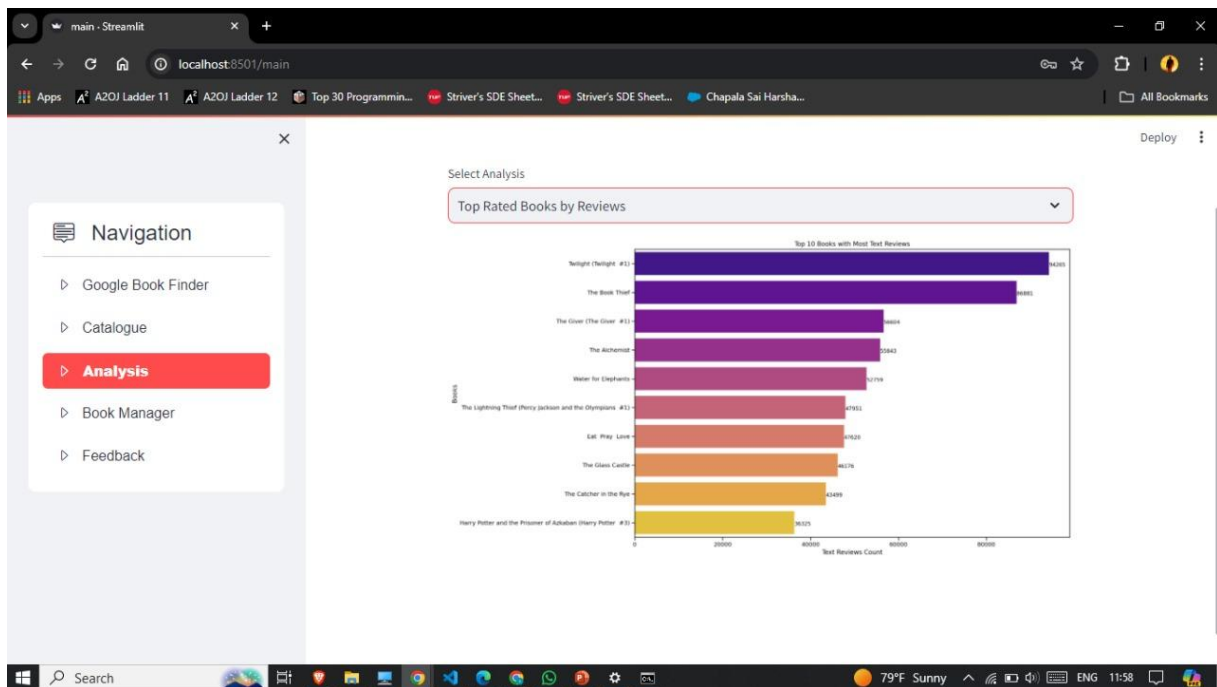
This is Backend Database in SQLite



The catalog feature in the library management system maintains a database of books, storing essential details like title, publisher, and rating stars. This database enables administrators to add, update, and delete books, ensuring efficient management of the library's collection. Users benefit from easy access to book information, allowing them to search for specific titles, publishers, or books with particular rating stars. The system's functionality includes data validation to maintain accuracy and integrity within the catalog, providing a streamlined experience for both administrators and users in managing and exploring books within the library.

4. ANALYSIS:





Top Rated Books Analysis:

This analysis identifies and presents the library's highest-rated books, based on user ratings. By sorting books according to their ratings count, users can easily discover the most esteemed and recommended books.

Most Occurring Books Analysis:

This analysis showcases the books that are most frequently borrowed or checked out from the library. By counting book occurrences in borrowing records, the system reveals which books are in high demand, aiding in resource management.

Top 10 Authors with Most Books Analysis:

This analysis highlights prolific authors whose works have a significant presence in the library's collection. By tallying the number of books attributed to each author, users can explore books based on their favorite authors' prolific output.

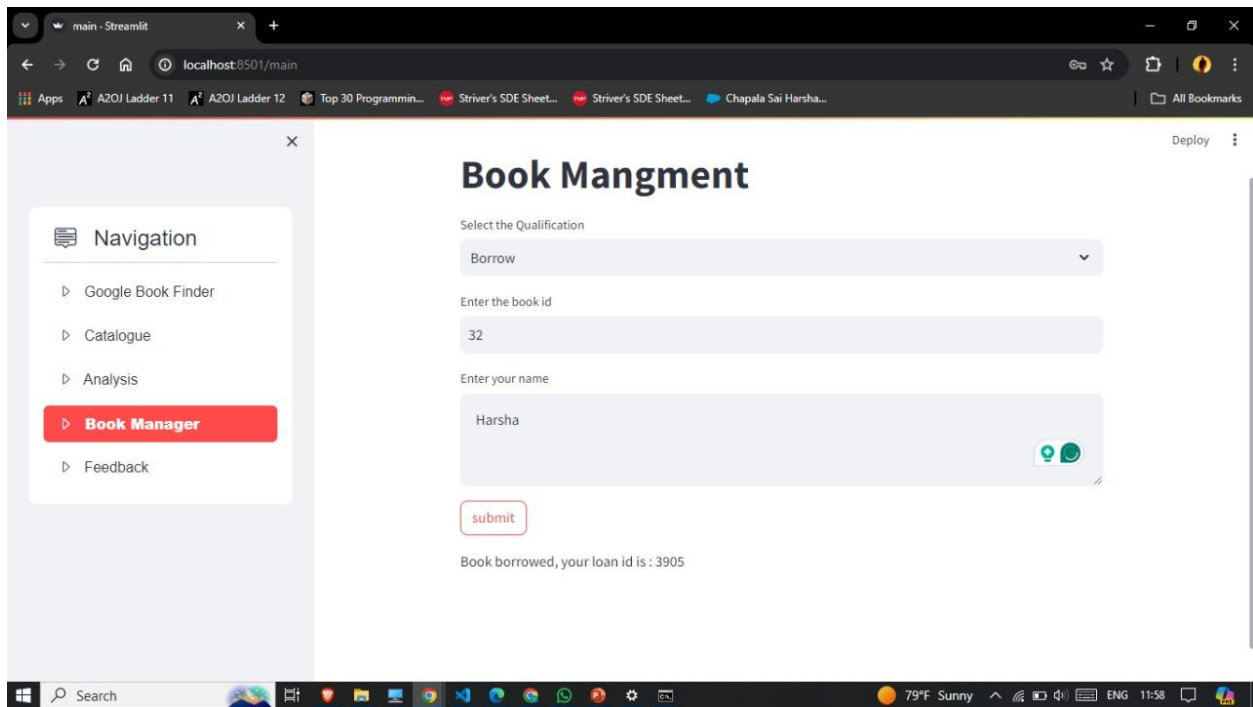
Top Rated Books by Reviews Analysis:

This analysis ranks books based on their text reviews count, offering insights into books that have garnered substantial attention and feedback from readers. Users can discover critically acclaimed or widely discussed books through this analysis.

Top Book Publishers Analysis:

This analysis focuses on publishers by counting the number of books published by each publishing house in the library's collection. It emphasizes publishers with a notable presence, enabling users to explore books from renowned publishing entities.

5. BOOK MANAGER:



The screenshot shows a web browser window with the URL `localhost:8501/main`. The application is titled "Book Mangment" (sic). On the left, there is a navigation sidebar with the following items: "Google Book Finder", "Catalogue", "Analysis", "Book Manager" (highlighted in red), and "Feedback". The main content area contains a form for borrowing a book. It includes a dropdown menu for "Select the Qualification" with "Borrow" selected, a text input for "Enter the book id" containing "32", and another text input for "Enter your name" containing "Harsha". A "submit" button is located below the name input. At the bottom of the form, a message states "Book borrowed, your loan id is : 3905". The browser's taskbar at the bottom shows various icons and the system clock indicating 11:58 on a sunny day at 79°F.

Borrowing Books:

Users enter their name and select their qualification (like student or teacher).

They then enter the book's name they want to borrow and click "submit."

The system records the checkout date and return date for the borrowed book and stores this information in a file named "BookManager.csv."

Returning Books:

Users provide their name and the book's name they want to return.

The system updates the return date for the returned book in the "BookManager.csv" file.

6.FEEDBACK

The screenshot shows a web application interface for providing feedback. On the left, a navigation sidebar lists several options, with 'Feedback' being the active and highlighted one. The main content area contains a form with four input fields: a text field for 'username' containing 'Harsha', a text field for 'bookid' containing '1', a large text area for 'feedback' containing 'This is a Great book', and a 'rating' field. A 'Deploy' button is located in the top right corner of the application window. The browser's address bar shows 'localhost:8501/main'. The Windows taskbar at the bottom indicates the system time is 11:59.

Inputting Feedback:

Users provide their name and feedback message using the designated text input fields. They also have the option to rate the book using a star-based rating system.

Submitting Feedback:

After entering their feedback and rating, users click the "Submit" button.

The system then stores the user's name, feedback message, and rating in a file named "feedback.csv."

Confirmation Message:

Upon successful submission, users receive a confirmation message confirming that their feedback has been recorded.

Feedback Record:

The feedback data, including user comments and ratings, is securely stored in the "feedback.csv" file for future reference.

PROBLEM ANALYSIS

Data Management Challenges:

Efficiently storing book-related information like titles, authors, publishers, ratings, availability status, borrower history, and feedback.

Ensuring data integrity, normalization, and optimal database querying performance.

User Management Issues:

Implementing secure user authentication and authorization for administrators and regular users.

Managing user-specific functionalities such as book borrowing, returning, feedback submission, and administrative tasks securely.

Streamlit Challenges:

Limited UI customization options can hinder design flexibility and customization of the interface.

Building complex UI components like interactive charts, dynamic forms, and data grids may necessitate additional workarounds or custom integrations.

Flask Challenges:

Designing clear and efficient URL routing patterns and views for handling HTTP requests and rendering responses.

Implementing data handling functionalities such as validation, sanitization, and serialization/deserialization of JSON or form data.

FUTURE SCOPE

Mobile Application Development: Designing a mobile app for the library management system would greatly improve user accessibility, catering to individuals who prefer accessing library services on their smartphones or tablets. The app would enable users to browse the catalog, reserve or renew books, view account information, and receive notifications on due dates or upcoming events, ensuring a seamless experience across various devices.

Enhanced Web Accessibility: Improving web accessibility ensures that all users, including those with disabilities, can easily navigate and utilize the library's online platform. This involves adhering to accessibility standards such as WCAG (Web Content Accessibility Guidelines) to make the website usable for people with diverse needs. Features like compatibility with screen readers, keyboard navigation, alternative text for images, and customizable font sizes and color contrasts contribute to a more inclusive browsing experience.

Data Analysis on Books Feedback: Implementing data analysis on user feedback provides valuable insights into the popularity and reception of different books in the library's collection. By collecting and analyzing feedback data, including ratings, reviews, and comments, the library management system can identify trends, preferences, and areas for improvement. This data-driven approach informs decisions on book acquisitions, collection development, and personalized recommendations.

Personalized Book Recommendations: Utilizing user behavior and preferences, the library management system can offer personalized book recommendations. By analyzing factors such as browsing history, borrowing patterns, favorite genres or authors, and user ratings, the system generates tailored recommendations aligned with each user's interests. This personalized recommendation engine enhances user engagement, satisfaction, and the overall browsing experience, encouraging users to discover new titles tailored to their tastes.

CONCLUSION

The document's ending talks about how useful the library system is for both the people who run the library and the people who use it. For those in charge, they can do things like keep track of books being borrowed and returned, update the library's information, and get helpful insights using advanced technology. This makes their job easier. People who use the library can easily find books, renew the ones they've borrowed, and access digital stuff from their phones or computers. This makes using the library more convenient.

The main goals of the system are to make the library experience better for everyone and to make running the library smoother. By giving the people in charge good tools and making it easy for users to find what they need, the system hopes to make the library a great place for everyone.