LibraDB - Library Management Platform

Library Management System using Flask and SQLite

Course Project Report

Group Members:

Aawaiz	22i-0845
Taha Khan	22i-2335

November 26, 2025

Contents

1 Introduction

LibraDB is a Flask + SQLite based library management system designed around role-aware access control and clean, modern web application patterns. It provides a lightweight, self-hosted platform for managing library resources, tracking book lending, and collecting user feedback in the form of ratings and comments.

At its core, LibraDB handles member registration and login using a single, unified authentication flow. Every user account is associated with a role (member or librarian), and librarian approval is required before a member can actively use the system. This approval gate, combined with secure password hashing and unique credentials, ensures that only verified users can access the catalog and booking features.

The platform supports comprehensive catalog management, including books and categories with unique identifiers. It tracks book availability, total copies, and related metadata, while offering features such as booking requests, librarian approvals, return tracking, and fine calculation. On top of this transactional flow, LibraDB allows members to submit ratings and comments for books, and the system computes average ratings in code to provide quick insight into user satisfaction.

From an engineering perspective, LibraDB showcases best practices for small-to-mid scale web applications: it uses SQLAlchemy for ORM, Alembic for database migrations, and an idempotent seed script to bootstrap demo data. This makes it not only a practical tool for libraries, but also a clear reference implementation for developers learning Flask, relational modeling, and role-based access control.

2 Targeted Audience

The primary audience for LibraDB includes:

- **Small to mid-size libraries** such as departmental, school, community, or organizational libraries that require a focused, lightweight solution to manage lending and catalog browsing.
- **Teams or organizations** that need a self-hosted lending and review workflow, where control over data and infrastructure is important.
- **Developers and students** who are looking for a simple but realistic Flask/SQLAlchemy reference implementation that demonstrates authentication, role-based access control, database migrations, seeding, and common CRUD patterns in a clear and structured way.

By targeting both operational users (librarians and members) as well as technical users (developers), LibraDB serves as both a practical tool and an educational example.

3 Project Scope

The current scope of the LibraDB application covers core features for user management, catalog handling, book lending, and feedback collection. The scope of the project (for the current app only, expanded) is summarized below.

User and Role Management

- Member and librarian roles are supported, with role-aware access to different parts of the system.
- Librarian approval is required before registered members can sign in and interact with the library.
- A single login/registration flow is provided for all users, simplifying the onboarding experience.
- Credentials are unique and stored securely using password hashing.

Catalog Management

- The system maintains books and categories with unique ISBNs and unique category names.
- For each book, LibraDB stores the title, author, description, total copies, available copies, and timestamps.
- Application logic supports creating, listing, and updating books and categories, as well as viewing detailed information for each book.

Booking and Lending Workflow

- Members can submit booking requests, selecting start and end dates for their loans.
- Librarians review and approve bookings before they become active.
- The system tracks the full lifecycle of a booking, including:

  - Return status.
  - Return request flag.
  - `returned_at` date.
  - `fine_amount` for overdue or late returns.

- Book availability is automatically updated when bookings are created and when books are returned, updating `copies_available` accordingly.

Ratings and Feedback

- Members can submit ratings and comments for books they have interacted with.
- The system calculates and displays the average rating per book in the application layer.

Deployment and Data Bootstrapping

- LibraDB ships with Alembic migrations to set up and evolve the database schema.
- An idempotent seed script inserts:
  - Demo users, including at least one default librarian.
  - Example categories and books.
  - A sample booking.
  - Sample ratings for demonstration purposes.
- The seed script is designed to avoid duplicates by enforcing uniqueness and checking existing data.

Overall, the scope focuses on a robust, realistic subset of library management features, while staying small and maintainable enough for educational and small deployment scenarios.

4 Requirements

4.1 Functional Requirements

User Authentication and Authorization

- Users can register as members via a registration form.
- Users can log in using their email and password.
- Passwords are stored using secure hashing.
- Each user has a role attribute (member or librarian).
- Librarians are responsible for approving member accounts before they can log in.

Catalog Management

- Librarians can create, list, and update books and categories (full CRUD may be supported in code even if not exposed in all UI flows).
- Each book belongs to a category.
- Users can view book details, including:
  - Title, author, description.
  - Category.
  - Total copies and currently available copies.
  - Average rating.

Bookings and Lending

- Members can request a booking for a selected book with specified start and end dates.
- Librarians can approve or reject booking requests.
- The system tracks:
  - Booking approval status.
  - Return status (returned or not).
  - Whether a return has been requested by the member.
  - The actual `returned_at` date (if any).
  - A `fine_amount` integer for overdue or late returns.
- `copies_available` is decremented when a booking is approved and incremented when a book is returned.

Ratings and Reviews

- Members can leave ratings (scores) and optional comments for books.
- The system calculates and displays the average rating per book.
- Ratings are associated with both the user and the book.

Administrative Safeguards

- Enforce unique email addresses for users.
- Enforce unique ISBNs for books.
- Seed script avoids inserting duplicate data when run multiple times.
- Deleting a parent record (such as a book or user) triggers cascading deletion of dependent bookings and ratings, ensuring referential integrity.

4.2 Non-Functional Requirements

Security

- Passwords must never be stored in plain text; they use hashing with a secure algorithm.
- Role-based access control is enforced (librarian vs member).
- Account approval is required before granting access to library features.

Data Integrity

- Database-level unique constraints on:
  - `users.email`
  - `categories.name`
  - `books.isbn`
- Foreign keys are used with cascading deletes where appropriate, to prevent orphaned rows.
- Reasonable default values are defined for boolean fields and counters (e.g., `approved`, `returned`, `copies_total`, `fine_amount`).

Reliability

- Database migrations are managed via Alembic to ensure consistent schema evolution.
- The seed script is idempotent and can be run multiple times without corrupting data.

Usability

- A single, unified login/registration flow simplifies onboarding.
- Demo credentials are provided to quickly access the system for testing or demonstration.
- Interfaces are kept straightforward, focusing on core tasks such as browsing books and managing bookings.

Maintainability

- The project uses a Flask app factory pattern.
- SQLAlchemy models are clearly defined and grouped logically.
- There is a clear separation of concerns: configuration, routes, models, and seeding are structured into dedicated modules.

5 ER Diagram

This ER diagram highlights the core entities and associations that support LibraDB's booking and rating workflows.

6 Normalized Schema (3NF)

The LibraDB relational schema is designed to be in Third Normal Form (3NF), reducing redundancy and ensuring data integrity.

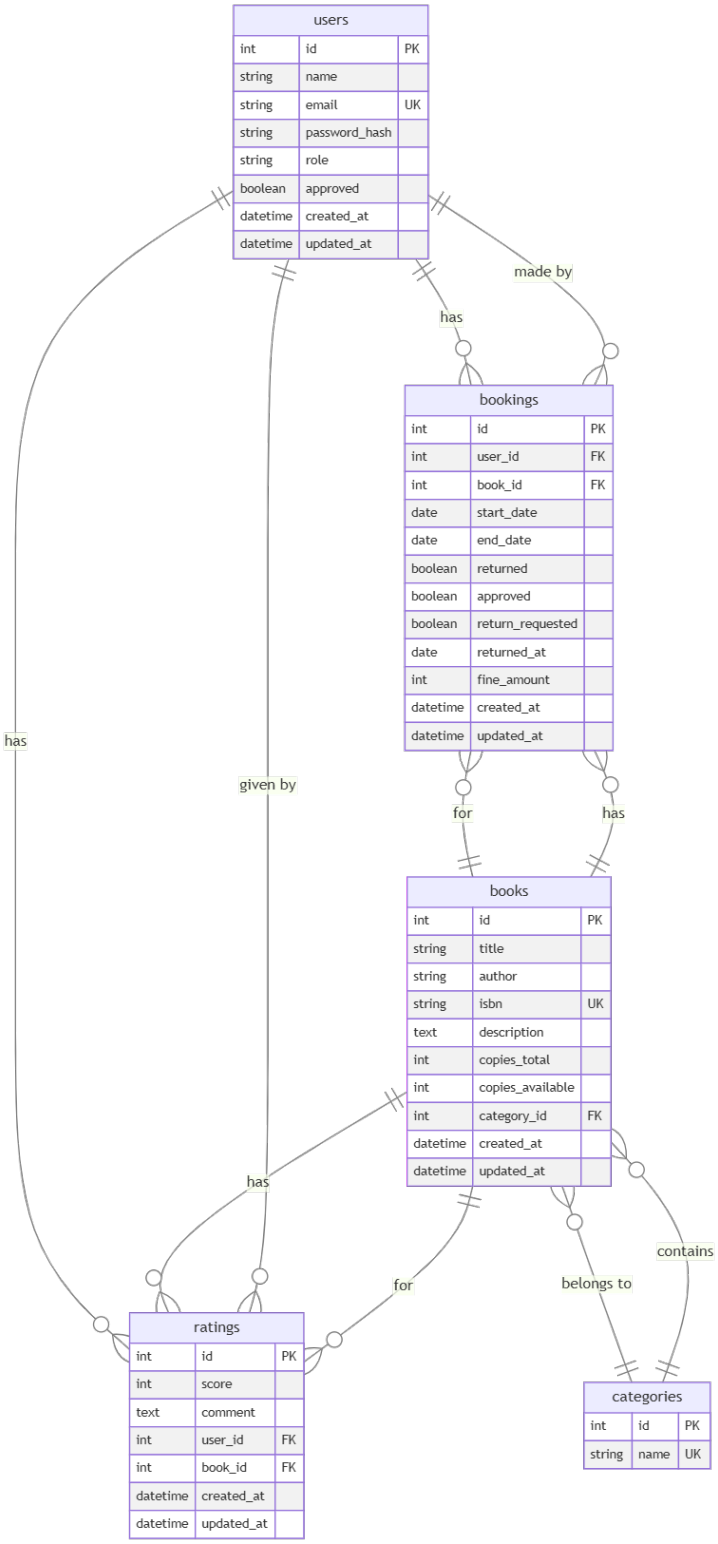

Figure 1: Entity–Relationship Diagram for LibraDB

`users` Table

Attributes:

- `id` (PK)
- `name`
- `email` (unique)
- `password_hash`
- `role` (default `member`)
- `approved` (boolean, default `false`)
- `created_at`
- `updated_at`

Relationships:

- One-to-many with `bookings` (`bookings.user_id` → `users.id`).
- One-to-many with `ratings` (`ratings.user_id` → `users.id`).
- Deleting a user cascades to delete their bookings and ratings.

`categories` Table

Attributes:

- `id` (PK)
- `name` (unique)

Relationships:

- One-to-many with `books` (`books.category_id` → `categories.id`).
- Deleting a category deletes its books and any dependent bookings/ratings via cascading relationships.

books Table

Attributes:

- id (PK)
- title
- author
- isbn (unique)
- description
- copies_total (default 1)
- copies_available (default 1)
- category_id (FK to categories.id)
- created_at
- updated_at

Relationships:

- One-to-many with bookings (bookings.book_id → books.id).
- One-to-many with ratings (ratings.book_id → books.id).
- Deleting a book deletes its associated bookings and ratings.

bookings Table

Attributes:

- id (PK)
- user_id (FK to users.id)
- book_id (FK to books.id)
- start_date
- end_date
- returned (boolean, default false)

- `approved` (boolean, default `false`)
- `return_requested` (boolean, default `false`)
- `returned_at` (nullable date)
- `fine_amount` (integer, default 0)
- `created_at`
- `updated_at`

Semantics:

- Each booking links one user to one book over a specified period.
- Lifecycle is captured via approval and return flags, as well as timestamps and fines.

`ratings` Table

Attributes:

- `id` (PK)
- `score` (integer, not null)
- `comment` (text, optional)
- `user_id` (FK to `users.id`)
- `book_id` (FK to `books.id`)
- `created_at`
- `updated_at`

Semantics:

- Each rating represents one user's evaluation of one book.
- Supports computation of average scores per book in application logic.

Because non-key attributes in each table depend only on the key (and not on other non-key attributes or partial keys), the schema satisfies 3NF, reducing redundancy and update anomalies.

7 Conclusion

LibraDB demonstrates how a focused, role-aware web application can effectively support the day-to-day operations of a library while remaining simple and maintainable. By clearly separating concerns between authentication, catalog management, booking workflows, and feedback collection, the system provides a cohesive experience for both librarians and members.

On the technical side, LibraDB showcases modern best practices in Flask application design: a normalized relational schema, SQLAlchemy models, migrations with Alembic, and an idempotent seed process for reproducible environments. These choices make the project suitable not only for production use in small to mid-size libraries, but also as a teaching and learning tool for students and developers exploring web development and database design.

Looking forward, LibraDB can be extended with additional features such as fine payment integration, notification systems (email or in-app), advanced reporting dashboards, and richer search and filter capabilities. Even within its current scope, however, LibraDB provides a robust and elegant foundation for managing library resources and lending workflows in a secure, structured, and user-friendly way.