Author

Ranjeet Sharma
21f2001119
21f2001119@ds.study.iitm.ac.in
Student at IIT-Madras pursuing Bachelor's degree in Data Science and Application, with a fervent interest in development, and programming.

Problem Statement - **Library Management System** - V2

Description

This project is about an Online Library Management System. There will be one admin and many users. Users can sign up/register and can start reading books or issue e-books. Admin can perform CRUD Operations on Sections and Books and handle incoming book requests.

Technologies used

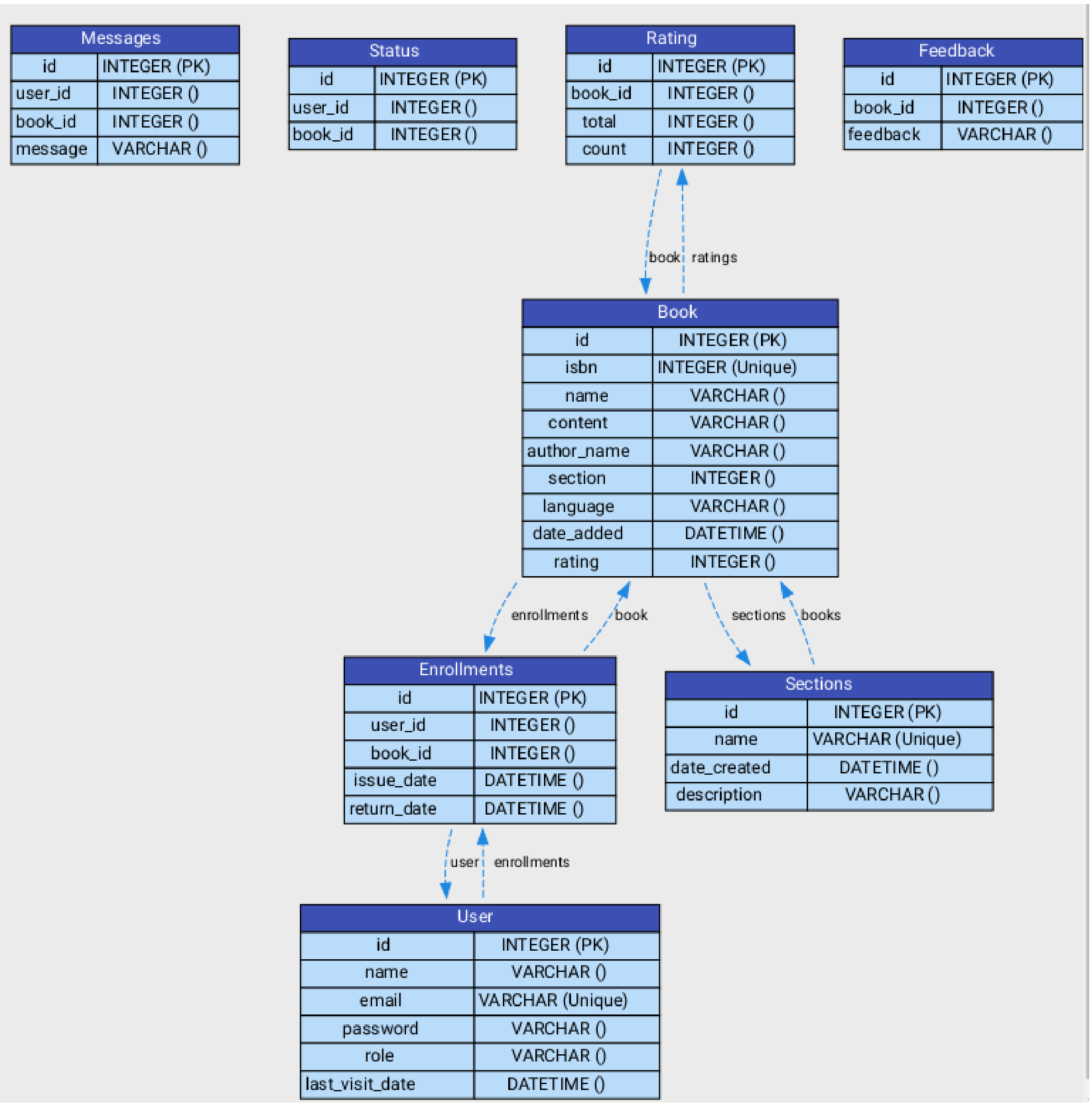
- **Flask**: For application code, to handle user requests, manage routing, and creating APIs.
- **Vue.js**: For building a dynamic and responsive user interface.
- **Flask-SQLAlchemy**: For interaction with the database.
- **Flask-Bcrypt**: For hashing passwords.
- **Bootstrap**: For quick CSS styling and aesthetics.
- **SQLite**: For data storage.
- **Redis**: For caching.
- **Redis and Celery**: For batch jobs and task queue management.

Architecture and Features

app.py: Contains the main code to run the Flask application. It initializes Flask and Flask-SQLAlchemy objects and contains necessary imports from controllers.
Controllers: Contains all the routing for the project.
models.py: Contains the schema for database design using Flask-SQLAlchemy. It includes classes that represent tables in the database, including columns and relationships between tables.
Static and Templates: The static folder contains global.css along with a few images, and all the HTML files are kept in the templates folder.

- **CRUD** Operations for **Sections** and **Books**.
- **Search Functionality**
  - ➔ Both admin and regular users can search based on book name, author, sections.
- **Bar Charts** also visible on the admin's dashboard for better track of books and sections.
- **Batch Jobs and Task Management**
  - ➔ **Redis and Celery**: Used for managing background tasks and scheduling periodic jobs like sending reminders and generating reports.

DB Schema Design



Messages	
id	INTEGER (PK)
user_id	INTEGER ()
book_id	INTEGER ()
message	VARCHAR ()

Status	
id	INTEGER (PK)
user_id	INTEGER ()
book_id	INTEGER ()

Rating	
id	INTEGER (PK)
book_id	INTEGER ()
total	INTEGER ()
count	INTEGER ()

Feedback	
id	INTEGER (PK)
book_id	INTEGER ()
feedback	VARCHAR ()

Book	
id	INTEGER (PK)
isbn	INTEGER (Unique)
name	VARCHAR ()
content	VARCHAR ()
author_name	VARCHAR ()
section	INTEGER ()
language	VARCHAR ()
date_added	DATETIME ()
rating	INTEGER ()

Enrollments	
id	INTEGER (PK)
user_id	INTEGER ()
book_id	INTEGER ()
issue_date	DATETIME ()
return_date	DATETIME ()

Sections	
id	INTEGER (PK)
name	VARCHAR (Unique)
date_created	DATETIME ()
description	VARCHAR ()

User	
id	INTEGER (PK)
name	VARCHAR ()
email	VARCHAR (Unique)
password	VARCHAR ()
role	VARCHAR ()
last_visit_date	DATETIME ()

API Design

- ➔ **Book Management API**, **Section Management API**, **Enrollments Management API**, **Profile Management API**, **Search**, **User Authentication/Management API** and **Admin** relevant endpoints with **GET**, **POST**, **DELETE**, **PUT** methods.
- ➔ **Book Management API:** `GET /api/book/<book_name>, POST /api/book, DELETE /api/book/<book_name>, PUT /api/book/<int:book_id>`
- ➔ **Section Management API:** **`GET /api/admin/sec, POST /api/admin/sec, PUT /api/admin/sec/<section_name>, DELETE /api/admin/sec/<int:section_id>`**

Video

https://drive.google.com/file/d/1PkFru_y1Jb-TIRfNKdrwNj2mbwkmsMNH/view?usp=sharing