Author

Swetha Suravajjula
21f1002451
21f1002451@student.onlinedegree.iitm.ac.in
I am a 4rd-year student of Computer Science Engineering at Mahindra University and a BS student in Programming and Data Science student at IIT Madras. I enjoy learning new things and exploring the world of Data Science. I aspire to become a Data Scientist. I also enjoy singing western and Carnatic music.

Description

The aim of this project is to create an e-library management application with an admin account and a general user account using backend flask apis and vue.js in the frontend. This project also uses celery , celery-beat and redis for backend jobs in order to send monthly reports, daily remainders and book access revoke via mail.

Technologies used

The main technologies and libraries have used to create this application are:

Flask: It is used to create an application and build Flask-APIs in the backend

Sqlalchemy: It is used to simplify database interactions with the flask app.

Celery: It is used to create backend jobs for sending emails about monthly report,daily reminders and automatic book revoke access.This was mainly done using celery-beat

Redis: this is used as a message broker in celery

ThunderClient: This is used to testing my backend apis

Vue js: This is the frontend part of my application where it renders the data received from the backend api

DB Schema Design

My database has eight tables.

1) User Table
   Columns: Id,username,email,password,active,fs_uniquifier
   - Many to many relationship with roles through the 'roles-user' association table
   - One-To-many relationship with activites through the UserActivity Table
2) Role Table
   Columns: id, name, description
   - Many-to-Many relationship with the users
3) User-Role Table
   - Columns: id, user_id,role_id
   - This is an association table
   - many-to-many relationship between users and roles

4) UserActitivity Table

- Records the user activities within the application
- Columns: id,user_id,activity_type,activity_timestamp
- foreign key : user_id

5) Section Table
- Sections of the library
- Columns: id,name,date_created,description
- One-to Many relationship with books

6) Book Table
- Books of the library
- Columns: id, name, book_pic,content,authors, date_issued, return_date, section_id, total_copies,available_copies, no_of_pages,date_created
- foreign key: section
- One-to-many relationship with issues and feedback

7. Issue Table
- Records the books issued to the user
- Columns: id, user_id, book_id, section_id, issue_id,issue_date, return_date
- foreign keys: User, Book, Section

8. Feedback Table
- Stores user feedbacks
-Columns: id, user_id, book_id, section_id, rating, comment, created_at
- foreign keys: User, Book, Section

Architecture and Features

I designed this web application in such a way that I have stored all my backend apis, configurations of the applications into a single Python file called **app.py.** I have imported the database models from **model.py**. I have set up a celery configuration file in **wokers.py** and **celeryconfig.py.** I designed all my backend jobs in **tasks.py** and mail configurations in **mailservice.py.** These are all imported in **app.py** file.

I wrote all my backend APIs of user and admin in **app.py.** Login, register, viewing sections and books are the common APIs to both admin and user. For users, I have designed apis for returning the book, requesting a book, user cart for viewing the issued book.The interesting part of request-book api is that it automatically issues book to the user after checking whether the requested book has already been his/her active issued book, whether there are available copies of the book and also whether the user has exceeded the maximum no of issued books which is 5. There is also an automatic revoke access of the book from the user in case the user exceeds the deadline which is 7 days. I have implemented it as a backend job using celery.If the user wishes to return book beforehand; they can return it using the return book button and then provide a feedback. Coming to the admin; I have designed API's for CRUD operations on e-books and sections. The admin can also see user activity, feedbacks of the books , issued details , user profile and user details.

Video

https://drive.google.com/file/d/1Fz2tkdZQZwUaKqRKDLKAmtirgp3g2_GS/view?usp=sharing