Project Report

Modern Application Development - II

Author
Soham S. Kulkarni
22f3002597
22f3002597@ds.study.iitm.ac.in

Personal Introduction

I am a full time BS degree student, a tech enthusiast and an Indian classical music practitioner. My main interests include developing data-driven applications, developing AI/ML models and reading about new research in the field. Python is the basis for the majority of my work. This project was especially a lot of fun to work on, I learnt many new things and gained a lot of experience. There were many challenges that I had to overcome which was the most important part in my learning process.

Project Description

The project discussed below is a successor of the Bookit application developed as a part of Modern Application I. The project discussed below aims to provide an interface for an e-library (i.e. a Library Management System). This application can be used by both users and librarians. Most of the functionalities remain the same as the previous version wherein the user can request, read and return a book (a book can be accessed for a maximum of 7 days after which access will be revoked), buy an e-book for a price, cart functionality etc.In the case of a librarian, they can create (add), read, update, or delete books, sections, authors, etc. They can also view book-related statistics, approve or reject book requests, and revoke access to issued books. Functionality such as monthly report, on demand csv report for librarian and daily report for user have been added in this version.This version provides a better UX because of the javascript used. The app security is also improved with the usage of JWT tokens (Flask-JWT extended).In terms of code the new code has been split up into two different parts: backend and frontend. Backend is developed using Vue.js while the backend part still uses flask, the only difference is that now it uses flask blueprints which makes it easy to organize the code.

Technologies/Modules/Libraries used

Backend

Flask: An python library for developing backend of the web applications
Flask-SQLAlchemy: Flask extension used to integrate SQLAlchemy
Flask-Cache: Flask extension that adds caching support
Flask-JWT-Extended: Flask extension that allows token-based authentication
SQLite: It is a database engine
Celery: It is a distributed task queue for Python applications

Redis: Redis is an in-memory database used for caching and celery tasks/results

Frontend

HTML: Used for creating templates

CSS: Used for styling web pages

Bootstrap: A simple yet powerful framework used for styling web pages

Vue.js: An open-source front end JavaScript framework for building user interfaces

SweetAlert: A JavaScript library that allows developers to create beautiful alerts

Blueprints

To organize the backend routes into an easy-to-fetch way. Following are the blueprints that are used

Authentication: `name='auth_bprint', url_prefix='/auth'`

User: `name='user_bprint', url_prefix='/user'`

Sections: `name='section_bprint', url_prefix='/sections'`

Books: `name='book_bprint, url_prefix='/books'`

Librarian: `name='lib_bprint', url_prefix='/lib'`

Author: `name='author_bprint', url_prefix='/authors'`

Database design

Models:

As per the application requirements 5 different models have been created

User: Has various attributes for the user:

usr_id: User id of the user

f_name: First name of the user

l_name: Last name of the user

email: Email of the user

password: Password of the user (encrypted)

role: Role of the user

books_owned: Books issued to user (Many to many relationship with Book)

books_requested: Book requestes by user (Many to many relationship with Book)

cart: Cart details of the user (Many to many relationship with Cart)

Methods:

set_password: Encrypt and store the supplied password for the user

check_passw: Encrypt the given password, check against stored password

Book: Has various attributes for the book:

book_id: Book id of the book

book_name: Title of the book
book_description: Description of the book
content: Name of the file having the book content
section_id: Section id for the books section (Many to one relationship with Section)
price: Price of the book
feedbacks: Feedbacks about book (Many to many relationship with Book)
date_added: The date on which the book was added

Section: Has various attributes for the section:
section_id: Section id of the book
section_name: Name of the section
section_description: Description of the section
date_created: Date on which the section was created
books: Books that the section has (One to many relationship with Book)

Author: Has various attributes for the author:
author_id: To store author id of the author
author_name: To store name of the author
author_description: To store description of the author
books: To get the books that have been authored by the author

Cart: Has various attributes for the cart:
cart_id: Cart id of the cart
user_id: User id of the cart owner
book: Books in the cart

TknBlockList: Has various attributes about the blocked JWT's
tkn_id: Id of the blocked token
jti: (JWT ID) claim in provides a unique identifier for the token
created_at: It is the time when JWT was added to the blocklist

Relationship Tables:

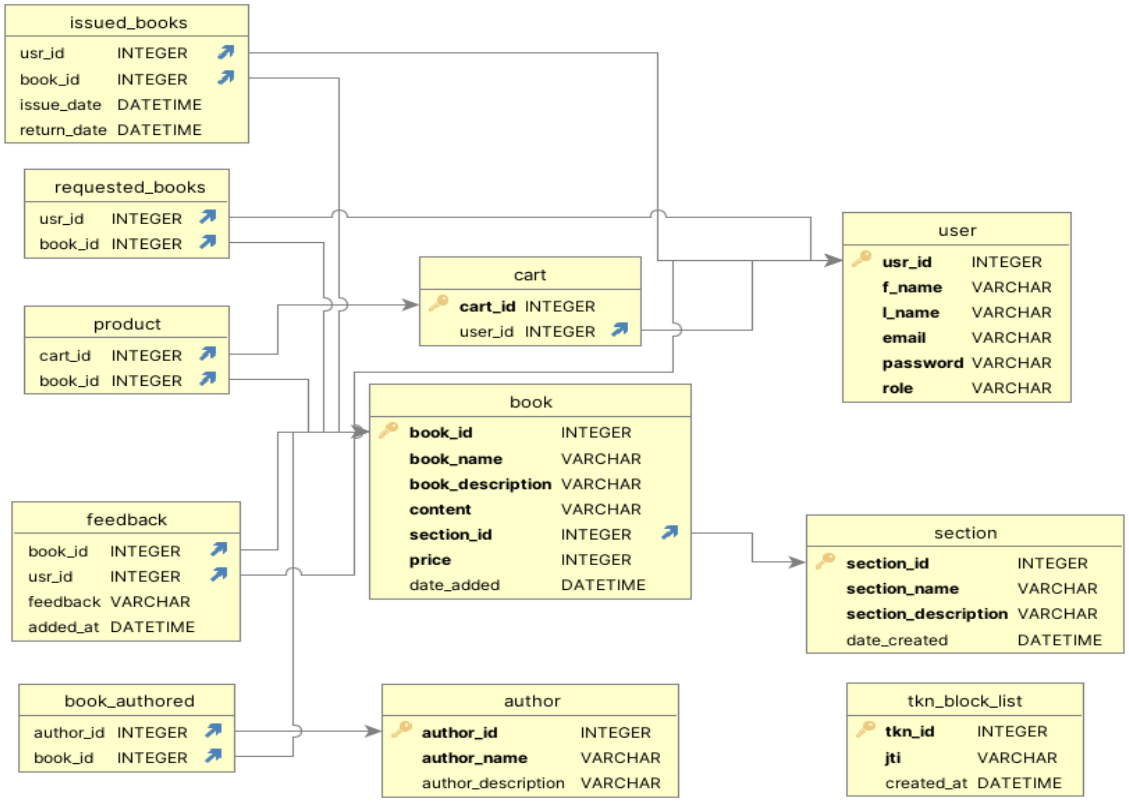
issued_books: This is a relationship table for many to many relationship between the book and user model. This relationship is useful because one user can have many books issued and vice-versa.
requested_books: This is a relationship table for many to many relationship between the book and user model. This relationship is useful because one user can have many books requested and vice-versa.

book_authored: This is a relationship table for many to many relationship between the book and author model. This relationship is useful because one book can have many authors and vice-versa.

product: This is a relationship table for many to many relationship between the book and cart model. This relationship is useful because one book can be in many carts and vice-versa.

feedback: This is a relationship table for many to many relationship between the book and user model. This relationship is useful because one book can have many feedbacks and vice-versa.

ER Diagram:



Video:

https://drive.google.com/file/d/1l8hefT5NJ_2w9OZtuNCzG7HTRecUnI_r/view?usp=sharing