Pustakalaya

Pustakalaya is a Library management app. It has great features for Book lovers. It can help the librarians manage the users easily. Users can take permission from the librarians to read the books for a specific amount of time. The database of the app has been made comprehensive, so that it has no lack of features. For developers who are looking to create a brand new library management app with the features of their choice, they can leverage the API to build on our already existing data in no time, without the need to do everything from scratch.

Inspiration

My goal from the beginning was to create an application which is beautiful not only from outside but also from inside. It meant the way I was going to organize the code had to be modular and to follow the effective design practices. Although minimalism of the features is what I usually aspire for, I'm aware there is no end to human desire for features and often design tastes evolve over time, so one of the other goals was to make such a design which would make adding new features or modifying the existing ones really simple, even for someone who has just forked my project and wants to get started without spending much time on understanding the already existing code.

Overall design

Overall design mostly follows the MVT architectural pattern. It has the following salient features.

- **Flask** - A micro web framework of Python is used in the application. Specifically with Flask, I'm using the Application factory pattern with Flask blueprints to make the application code modular.
- **Modularity** - For making different parts of the application REST API and controllers for the web application modular python packaging standards are followed to achieve a clean design and avoid duplication of code.
- **Utils** - All the logic in the application is there in a separate utils package so multiple parts of the controllers or API can use it without the need for duplication of code.
- **Templating** - Jinja is used as the templating system which receives data from the model with the help of controllers and renders itself as HTML. I've tried to make the templating use the inheritance features, macros, include etc to alleviate the requirement of writing long HTML files, which can lead to redundancy and many bugs.
- **Database** - Model is set up as a sqlite database. On the controllers side, SQLAlchemy ORM is being used for the database operations serving as the middleman between the model and controller.
- **Language support** - Application has support for multiple languages.
- **Aesthetics -** For making the aesthetics of the application appealing to use, I've made use of the Bootstrap framework, Material icons from Google. I've tried to follow Jakob Nielsen's usability heuristics for user interface design.

Database design

- **Managing users -** Since there are two types of users on this platform, Admins, which are the Librarians and Normal users, there are different ways we can solve this problem depending on the number of tables for the users and separating data between them, resulting in Single, Concrete and Joint table inheritances. I've chosen to use the Joint table inheritance, so I've a table which stores all the basic info common to every user, and then there are different tables which inherit from this base table of users to include the attributes specific to that role. This hierarchical method is helpful in avoiding many common problems like redundancy and lack of normalization.
- **Managing relationships between tables -** Appropriately all the relationships are handled either as one to one, one to many or many to one relationships since there are a lot of tables with relationships, along with handling the corner cases when a decision has to be made when all the items in one side of a relationship are removed.
- **Indexing -** Indexing has been set on various columns of different tables which can receive a high volume query to make the database operations fast, like the Feedback table, book table. since users can access these more frequently.
- **List of Tables -**
    - User
        - Librarian - Inherited from User
    - Author
    - Section
    - Book
    - Collection - For storing the list of books of users in their collection
    - Genre - For book genres within sections
    - Language
    - collection_books- Association table between books and collection table
    - book_author - Association table between author and book table for storing the many to many relationships between the tables
    - feedback - Association table between users and books for storing the feedback of users like ratings and comments by users