Manuel Tenorio Villagomez

Sonar Wish List App

May 19, 2020

Wish List Design Doc

Developing a Digital Management System

Abstract

The ask was to create a REST API in python3 that gives users the ability to manage their own book wishlist. The API would handle add, update, and delete, of the users books. These updates would then be made persistent with the use a of database.

Technologies

For this particular project Flask was the main framework used in order to deploy in a common and straight forward python framework. Since the app was developed using a mac OS sqlite3 was used as a database technology. Flask itself has a way to access sqlite, however it seemed to have too much boilerplate therefore it was decided to use a combination of Marshmallow and SQLAlchemy to support the connection to sqlite. Flask along with the supporting libraries are very well documented and easy to get started from scratch.

Architecture

The application has a standard MVC model architecture where it consists of a client sending request to the API layer and the API layer referring to a model in order to query the database and returns the result. As this is a simple application the API and UI routes where kept in the same file, but they could have been separated using blueprints. In order to communicate between client and API a standard JSON string will be passed as it is simple to read and universally used.

Note: Packages could be make to better organize the app if it where to be added on in the future.

UX UI Design

The client has only the necessary views to run a program such as this. With a navigation bar to guide the user around the website. A user can visit the websites home-screen, and about page without the need to login. In order to see the users book wishlist the user will be asked to first login, if no account has been created the user has the option to register themselves. No option to reset or retrieve password were set.

Database

The schemas for the databases are quite simple they simply hold the values of the object book and user along with a random generated id as a primary key. A third table was added in order to associate books with a user, this way we remove the need of duplicate books. The third table holds a randomly generated id as a primary key and two foreign key id's referring to the user and book. This relationship maps the books a user "owns".

Logic

A user is able to create a new book, this book will be shared with all the users that also add the book in order to minimize duplicate data. A user is able to retrieve all books that they have added themselves. The user cannot see books that they have not added. For this version a user can update and delete any book that they have access too. Meaning that they are able to change a books attribute and even delete it which will affect all users that "own" or subscribe to that book.

Note: In a future iteration we should restrict the deletion to only remove the mapping from the Owner table. So it may not affect other users.

API (sample)

login(email, password) POST

This allows the client to authenticate the application by checking its credentials again the database.

register(firstName, lastName, email, password) POST

This allows the user to register to the application by and saved the information in the database.

book(id) GET

Will retrieve the one book which matches that particular id.

book(user) GET

Will retrieve all the books that the user currently has.

book(id) DELETE

Will delete the specific book and delete record from Owner table

book(id, title, author, isbn, date) PUT

Will update the specific book

Contentious Issues

- Flask vs Django
- Organizing the application using blueprints
- UI was not needed
- DB design could be different
- Have a test DB to make sure real data and mock data is not mixed
- Code review need to catch edge cases (exceptions)

Testing

For this application testing was done using python's unittest. For all book API calls and routes I have tested their REST code and returning data. This will ensure not only that the API call is successfully completing, but that it is returning the expected data. Due to time constraints some of the user API calls were not tested.

Technologies

- python3
- pip3
- flask
- sqlalchemy
- marshmallow
- wtforms
- requests
- visual studio code
- terminal
- chrome
- sqlite3
- bootstrap
- jquery
- html
- css