Wishlist API

Technology Used

- Python 3, SQLite - These were the recommended tools to start out with and build off of. The challenge was quite open ended for additional tools, so I went with the preferred

- Flask, SQLAlchemy - These tools allowed for quick prototyping and Fail-Fast testing. They are both well supported python modules that allow for building API hooks easily, as well as storing data easily

- Mockaroo - A handy website to help populate a database with large amounts of mock data.

- Pytest, requests - These tools allow us to test our API endpoints and quickly add additional tests to ensure they work properly for every addition in the future.

Design Choices

This application was designed around linking existing users to books. I implemented an additional relationship table (wishlist) to allow users to have one to many wish lists of varying names associated with their accounts.

Additionally, this implementation of the API covers the Model and Controller from MVC design philosophy. If desired in the future, views could easily be made for each end-point without altering API functionality.

Usage

There are 5 endpoints available to call

- "/data/wishlist/add" - A post method that adds an *wishlist entry* to a users *wishlist*, takes **book_id** and **list_id** as request arguments

- "/data/wishlist/get" - A get method that lists all currently stored *wishlists* along with their *wishlist entries* and *owners*

- "/data/wishlist/get/**wishlist_id**" - A get method that lists a *wishlist* at **wishlist_id** with its *entries* and *owner*

- "/data/wishlist/update/**wishlist_entry_id**" - An update method that updates the wished for *book* at **wishlist_entry_id.** It updates it the *entry* record with the request argument **book_id**

- "/data/wishlist/delete/**wishlist_entry_id**" - A delete method that deletes a *wishlist entry* at **wishlist_entry_id**