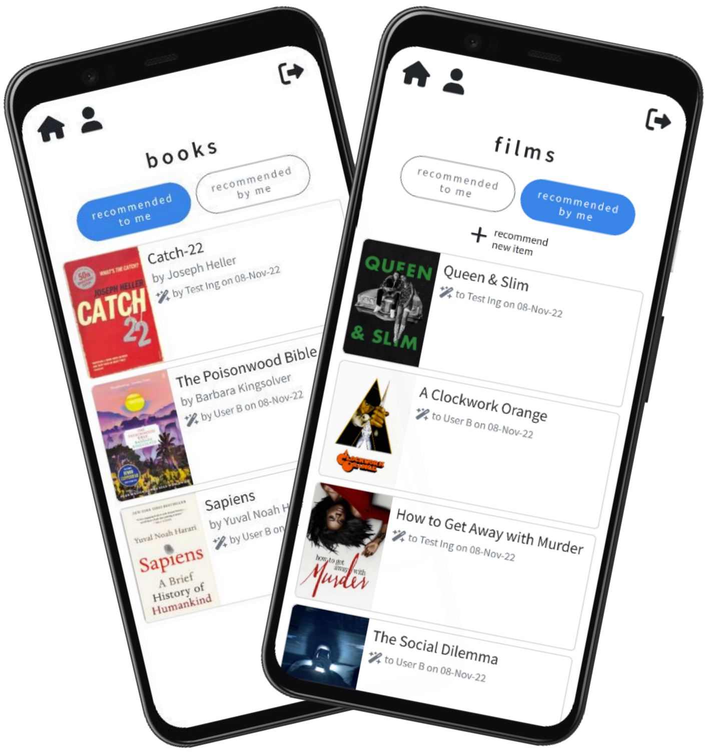
Recommendation App

Software Documentation



Documentation for code at
https://github.com/clairesquires99/RecommendationApp

Author: Claire Squires
Date: 15/11/2022

Table of Contents

Introduction

This recommendation app 'Recommenda' is a space for friends to recommend various types of media to each other. Users can sign up by creating a profile, and then login to start following other users and recommending items to their friends. This app allows users to keep all their recommendations for books, films, music, podcasts and articles in one place.

A demo of the app is hosted at https://clairesquires99.pythonanywhere.com/ (live at the time of writing).

Requirements

Functional

- Users are required to login to use the app
    - Email address
    - Password
- Users follow other users
- Users can only recommend an item to a user that is following them[1]
- Users can recommend an item to any number of the people following them
- The types of items include
    - Books
    - Films (movies and TV series)
    - Music
    - Podcasts
    - Articles

Non-functional

- Content and functions must load in a timely fashion so as not to irritate users with long loading times
- The app should support concurrent users

Design Process

This app is being developed using the agile development process. Each new iteration implements a new iteration and aims to be functional and deployed.

User Interface Design

The interface design aims to be uncomplicated and easy to use. This design choice was motivated by two reasons: first, a pleasant user experience, and second it allowed the implementation to focus on function rather than aesthetics. This app is primarily intended for use on mobile devices, and as such was designed using the mobile-first design principle. The app however, is fully responsive, and can therefore be rendered appropriately on devices of any type or size.

[1] Allows popular people (e.g. Barack Obama) to recommend items to all his followers, without any of his followers (potential strangers) being able to recommend content to him

To achieve a simple as well as visually appealing interface, the Bootstrap (v5) framework is used in combination with a small custom CSS stylesheet.

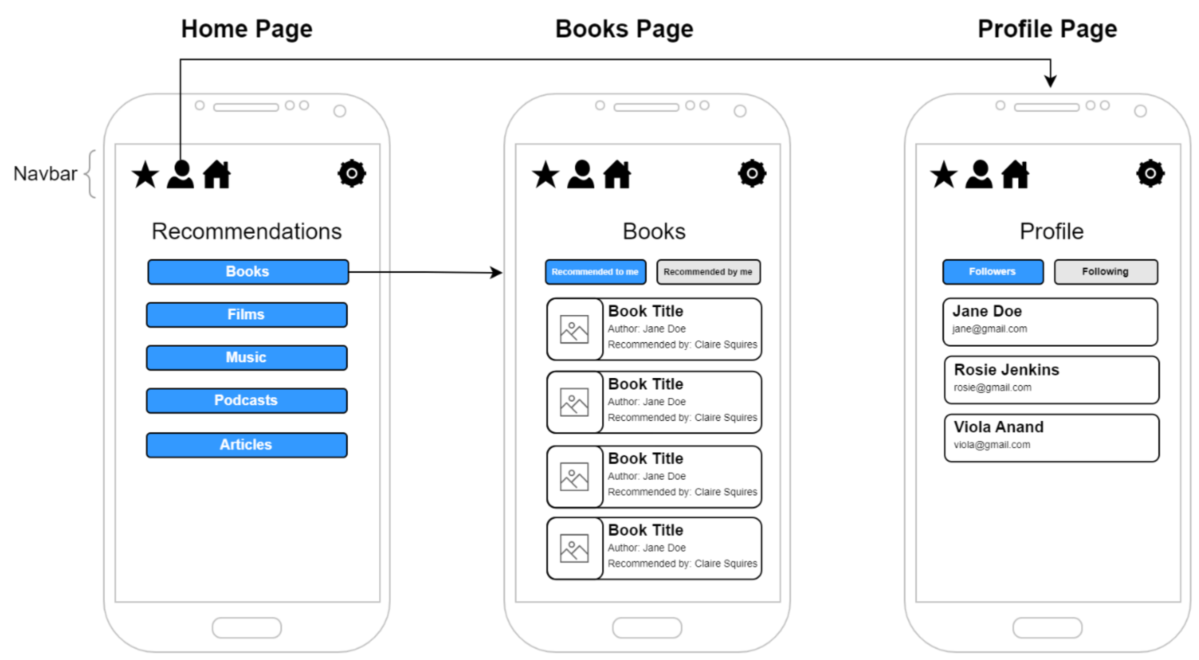


Figure 1 - Wireframes of UI

System Architecture

The app is implemented in Python using the Flask framework and follows the traditional Flask architecture. The `__init__.py` file creates the app, and further `.py` files contain the python code for the functioning of each of the views (`auth.py` and `main.py`). The directory templates contain the HTML files for each page. Most `.html` files extend one of the two base templates: `base_login.html` and `base.html`. The static directory contains the custom CSS styling, in addition to other static resources (images).

In addition, it makes use of blueprints to factor the application into two parts: the login system (`auth.py`) and the main system (`main.py`).

Two main parts of functionality were implemented: the social network, and the interaction with the APIs. The social media network allows users to sign up and then login using an email address and password, after which they can follow other users. Once the user has followers, they can recommend items to these followers. The APIs allow users to access content to recommend to their followers. Various APIs are used to search and fetch content of a specific media type.

Objects

The information required for the functioning of the app is treated like an object (although it is not strictly a defined Python object). These objects are detailed below:

- Users
  - First name
  - Last name
  - Email

    - Password
    - Recommendations
        - Books
        - Films
        - Music
        - Podcasts
        - Articles
- Item
    - Title
    - Author/artist

Data

The app requires data to be stored in a database to be accessed and presented to the user. The data is stored in a relational database using SQLite. The database is created using the schema in `models.py`. Operations on the data are performed using Flask-SQLAlchemy. This data consists of the objects highlighted in the previous section. See the specific attributes and relationships in Figure 2. The ID in each table is the primary key. The media type ID (e.g. Book ID) is the ID used in the respective API (denoted using green notes in Figure 2).

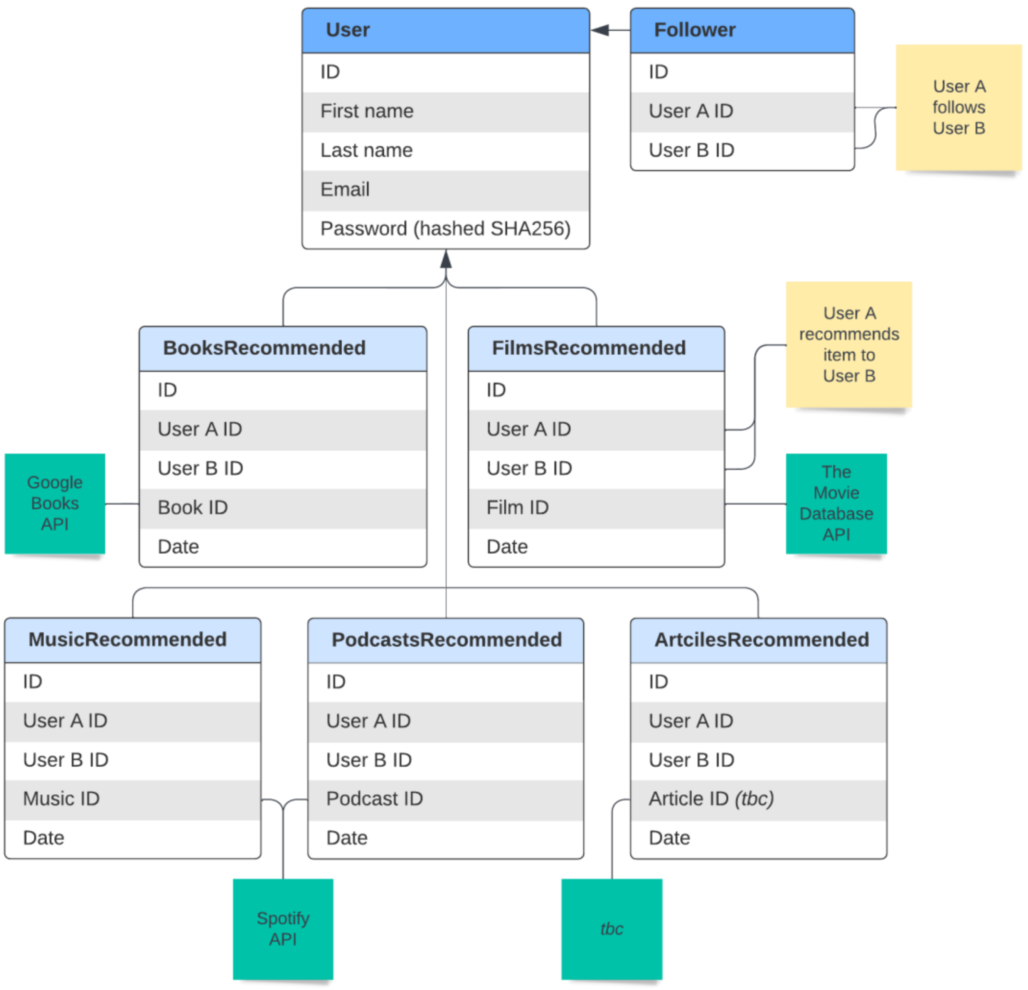


Figure 2 - Database schema diagram

A design choice was made to store the recommendations for each media type in a separate table (see Figure 2). This decision was motivated by the desire to reduce the time taken to search each table (since the media would already be provided).

As this project is currently in development, the structure and functioning of the `articles` table are still to be confirmed.

Testing

At present, no tests have been written for the app. While it is best practice to write tests before implementation, due to the personal nature of the project the priority was to implement an app (even if it does not work perfectly).

Continuing Work

At the time of writing, several features still need to be implemented.

For completion

- ☒Media type Books (released v1)
- ☒Media type Films (released v2)
- ☐Media type Music
- ☐Media type Podcasts
- ☐Media type Articles
- ☐Fix problems with unreadable titles from Google Books API (causes internal server error)

For improved user experience

- ☐Allow users to recommend items to themselves
- ☐Lazy loading on images
- ☐Allow users to recommend an item to several people at once
- ☐Allow users to delete recommendations
- ☐Allow users to unfollow users they are currently following