Interactive Book Search and Library System Inspired by Goodreads

Programming with Advanced Computer Languages

Language: Python

Authors

Moritz Fässler - 17-868-209,

Adele Bortoletti - 20-993-945,

Kyung Won Choi - 24-608-358,

Luis Pereira Baptista - 21-612-080,

Zhuojia Yin – 24-606-899,

Mingwei Wang - 23-601-255

University of St Gallen

Table of Contents

1. Project Description
2. Project Structure
3. Running the Application
4. Limitations of the Implementation
5. Learning Outcomes

1. Project Description

Inspired by Goodreads, this project focuses on building a searchable book database with personal library management and filtering capabilities. The objective was to replicate core functionalities of Goodreads in a simplified manner, tailored to the course's requirements. To achieve this, the team acquired and cleaned a book dataset from Kaggle, which formed the basis for developing robust backend functionalities. These included data handling, runtime loading, and filtering based on multiple criteria.

The user interface, developed using Streamlit, offers an interactive and user-friendly experience, allowing users to search for books, add them to their personal library, and track their reading status. The seamless integration of the front-end with backend modules ensures efficient data processing and dynamic interaction. GitHub served as the version control platform, facilitating collaboration and enabling parallel development to manage tasks effectively.

2. Project Structure

The project is organized into a modular structure to ensure clarity and efficient collaboration. At the root level, the directory contains a data folder for storing datasets, a `src` folder for core modules, and other essential files such as `app.py` for the main application logic and `requirements.txt` for dependency management. The detailed folder structure is illustrated below:

```
root
├── data
│   └── books_with_unique_isbns.csv
├── src
│   └── data
│       ├── data_loader.py
│       └── filters.py
├── app.py
└── requirements.txt
```

The dataset, sourced from Kaggle, contains book data from Goodreads. Before use, it underwent cleaning to remove incomplete or nonsensical entries. Placeholder ISBNs (e.g., ‘9999...’) were replaced with randomly generated valid ISBNs, ensuring uniqueness while adhering to standard formatting conventions.

The `data_loader.py` module is designed to read the dataset, stored in CSV format, and return it as a Pandas DataFrame. Special care was taken to ensure cross-platform compatibility, enabling seamless usage across different operating systems within the team.

The `filters.py` module provides functionalities for filtering the dataset based on user-defined criteria. It adopts a class-based structure with static methods, each dedicated to a specific filtering task. This modularity promotes maintainability and scalability. Example filters include searches by title, author, rating, genre, page count, and ISBN.

The main application logic resides in the `app.py` file, built using Streamlit. Key functionalities include:

1. **Session Management**: The application utilizes Streamlit's session state to persist user interactions, such as tracking selected books and storing search results. This ensures that user data remains consistent across reloads.
2. **Pagination**: To optimize performance, the application limits the number of displayed search results per page. While this improves clarity and speed, further refinement is needed to address cross-page functionality.
3. **User Interface**: The application's interface is divided into two sections. The library management section allows users to manage their personal library, including updating reading statuses and removing books. The search interface provides robust functionality for discovering books through various criteria, such as title, author, rating, and genre.

3. Running the Application

To run the application, users must first install the dependencies listed in the `requirements.txt` file. Once the dependencies are installed, the application can be launched by executing the following command in the project root directory:

```
streamlit run app.py
```

If a new browser tab does not open automatically, the application can be accessed manually via `http://localhost:8501`. Users are advised to ensure that Python version 3.8 or higher is installed and that Streamlit is correctly set up in their environment. Troubleshooting steps, such as verifying installations, can help resolve any issues encountered during setup.

4. Limitations of the Implementation

Despite its strengths, the project has certain limitations. One notable issue is the incomplete functionality of pagination across different slices of search results. While the current implementation improves performance and clarity, further refinement is needed to make pagination state-aware and more seamless.

Another limitation pertains to session management. Adding a book to the library triggers a full application reload, causing the user's current view to reset. This disrupts the user experience, especially when multiple books need to be added from the same search results. Implementing asynchronous updates, such as AJAX, could address this issue and enhance interactivity.

Finally, the project's scope constrained the inclusion of features such as user reviews and collaborative recommendations. While these functionalities would significantly enrich the application, they were deemed beyond the course's requirements and the team's available resources.

5. Learning Outcomes

This project provided valuable insights and skill development in both technical and collaborative domains. From a technical perspective, the team gained experience in data preprocessing, modular programming, and building interactive user interfaces with Streamlit. The project also required the implementation of state management and optimization techniques to address performance challenges.

From a collaborative standpoint, the use of GitHub facilitated effective version control and parallel development. Team members improved their ability to communicate and delegate tasks, ensuring that the workload was distributed equitably. The challenges faced during the project, such as ensuring cross-platform compatibility and managing application state, further enhanced problem-solving and critical thinking skills.

In conclusion, the project successfully demonstrates a simplified version of Goodreads with core functionalities for searching, filtering, and managing a personal library. Despite its limitations, the implementation lays a strong foundation for future enhancements and scalability.