bookchain

Frontend & Backend Integration

At this point, the frontend and backend of the application are fully integrated.

The mySQL database consists of 6 tables and is defined in **schema.sql**:

- **Users**: stores information about the users of the platform
- **Books**: stores information about books on the platform
- **User_books**: stores information about the relationship between users and books
- **Ratings**: stores information about ratings users have given books
- **History**: stores information about the physical movements of a book
- **Comments**: stores information about comments users have made on books

Following a Model-View-Controller structure, **models.py** – written around two major classes, User and Book, and a couple of helper functions – communicates with the database to save, retrieve and sort/filter information as necessary while **views.py** exchanges commands and information with the frontend (coded in javascript) through ajax calls.

It is probably easiest at this point to explain some of the provided functionality in user journeys.

- A user can find books she finds sitting unused through a search window in the application. All this information is pulled from the internet (Google Books API). Once she finds the right book (a book she owns), she clicks on a 'register' button which effectively causes this book as well as the user's relationship with the book to be stored in the database. Now this book will be part of the user's collection and prominently displayed on her dashboard as well. It will also be shown to other users logging into the service who can then decide whether they would like to request it or not.

- Every book has a book detail page with further information about the book. This information is retrieved from the database upon page load. On that page, users can leave comments and rate the book with 1- 5 stars – any of these actions will also be saved in the relevant database.

- Information from the database is almost exclusively queried on page-load. Every user sees a personalized dashboard with books and other information currently relevant to them. For example, once the user navigates to the dashboard, all books the user is currently reading, all she has uploaded, and all those she has requested will be displayed.