I implemented program using Java 11, Spring boot with web, jpa, h2, validation, and swagger dependencies.

As db, h2 used and created as a file automatically in the run classpath and with pre-populated data with a postconstruct bootstrap class.

For data persistance there are three classes under entities package. Also for api presentation a full and a summary (without collections) dto class created for each entity under dto package.

Swagger documentation used easy navigation and testing.

**Uris and files:**

Api root endpoint: <http://localhost:8080>

Swagger ui: <http://localhost:8080/swagger-ui>

H2-console: <http://localhost:8080/h2-console> (uri: jdbc:h2:file:./librarian)

librarian.zip: Exported intellij project.

Run.bat: to run program on a windows machine. For \*nix based OS same command can be run from console.

When program starts it creates db file on this directory.

Postman collection: Can be imported to postman for easy testing.

I decided to create 3 entities. Book, Author and Reader. There is also relation tables but omitted in this document.

**Entites:**

Book :

id : Long, auto increment

isbn: String, unique

title: String

shelf: String. Defines which shelf book is in.

line: Integer. Defines in which line book on given shelf

count: Integer. Number of books available to rent. Decreased when a book is rent and increased when returned.

price: Integer. Price that book will be rented.

authors: Set<Author>. Authors defined for this book.

readers: Set<Reader>. Readers currently rented this book.

Author :

id : Long, auto increment

name: String

books: Set<Book>. Books that author written.

Reader:

id : Long, auto increment

name: String

book: Book. Book that reader rented.

**Assumptions:**

Books need to have a unique isbn. Isbn must be 10-14 character long (validated through api)

Every book may have 0 or more authors. And 0 or more readers.

Every author may have 0 or more books.

Every reader may have at most one book rented at a given time.

A book cannot be deleted if there is at least one reader rented this book.

When a book is deleted, it is also deleted from authors’ book lists that is defined written by this author, so if an author has only this book, now has an empty book list, or if there are more books only this book is removed from set.

When an author is deleted same logic works as books.

During rent, first it is checked if the given reader already has a book rented, if so an exception is thrown. Also, if the book to be rented must have a count bigger than 0 else it is not allowed.

**Improvements:**

Here I listed a few shortcoming and possible improvements that could be made. Some are due to keep design simple for this small scale app and some due time limit.

Unit tests omitted due to time limitation

Could be implemented total money earned with rent and return apis. It is easy to implement with current design. But seems not necessary during design with time limitation.

A few more endpoints could be added like search by name (partial matches).

For error handling a general clas is used, exception throwing logic is considered as simple as possible and existing exceptions is used. This could be improved with custom or more suitable exceptions but omitted due to time and complexity concerns.