BookNest
Application for tracking your readings.
Your personal online library…

System Requirements Document
Multiplatform Application Development – DAM Project

Introduction

Purpose of the document

This document aims to help you create your own online library, where you can keep track of your reading, thus facilitating a count of the readings you have done both physically and digitally.

System scope

The system will allow the user to register, manage and track the readings they would like to have, are having or have already finished, being able to rate, give opinions and see their reading statistics both daily, monthly and yearly.

General description

The system will allow users to register, manage, and track their reading progress, including books they'd like to read, books they're currently reading, and books they've already finished. Users can rate books, leave reviews, and view their daily, monthly, and annual reading statistics.

Books can be added, placed on wish lists, and tracked as they progress, with the system tracking the reading time. Users can also mark books as they've read them, indicating they've stopped reading because they didn't like it, or that they've finished the book.

To add a book, users can search by entering the title; if it's already in the database, they can select it. Alternatively, they can submit a book by filling out all the fields before adding it to the database.

In general, you can rate your reading experience from 0 to 5 stars. To be more specific, you can add teardrop emojis if you cried, and other types of icons.

You can also leave comments about the reading.

Product perspective

The product will consist of a native Android application programmed in Kotlin using Jetpack Compose along with a RESTful API installed on an ASGI server with Python.

System functions

- User registration.
- User login.
- Edit user.
- Delete user.
- Reading log.
- Track your reading progress.
- Reading time statistics.
- Manage books you are currently reading.
- Manage books you have read.
- Manage books you want to read.
- Rate books.
- Comment on readings.
- Add books to the database.
- Search for books in the database.
- Abandon readings.

System users

- Users

Functional requirements

- **RF1:** The API, when deployed, finds the database to be empty.
- **RF2:** Once registered, all users will be able to access all API functions.
- **RF3:** The system allows the user to add books to the database.

- **RF4:** The system must allow entry to the system using a username and password.
- **RF5:** The system must allow the creation of a user with a name and password.
- **RF6:** The user can view, edit, and delete their lists.
- **RF7:** The user can view daily, monthly, and annual statistics.
- **RF8:** The user can rate and comment on the books.
- **RF9:** The user can leave a reading unfinished.
- **RF10:** The user can add books to their wishlist.
- **RF11:** The user can save the page or the percentage read of the book they are currently reading.
- **RF12:** The user can edit their profile.
- **RF13:** The user can delete their profile.

Non-functional requirements

- **NFR1**: The system consists of two distinct parts: a native Android frontend and a backend with FastAPI that will include a database.
- **NFR2**: The backend will be Dockerized.
- **NFR3**: The Android application will be written in Kotlin using Jetpack Compose.
- **NFR4:** The user interface should be intuitive and easy to use, with low loading times.
- **NFR5**: The system must comply with security standards to protect sensitive student information.
- **NFR6**: The system must be covered with tests and each new functionality must have its own tests, at least unit tests but also integration tests.

Interface requirements

The graphical user interface will be developed with Jetpack Compose, the framework for creating interfaces for Android applications.

System requirements

The Android application will be developed for Android mobile phones running Android version 12 or higher (API 31). It will use Kotlin and, as previously mentioned, Jetpack Compose.

The backend will use FastAPI, in its latest version available at the time of application development.

As for the database engine, the backend will use MySQL, in its latest version. You will need to use the SQLModel ORM, which is based on SQLAlchemy, uses Pydantic, and is well-integrated with FastAPI.

Finally, the backend must be deployed using Docker containers (at least two containers: one for FastAPI and one for MySQL).

Acceptance criteria

The system will be considered accepted if:

- All functional requirements have been implemented and tested.
- The application has been documented.
- Usability tests have been carried out with at least 5 teachers.
- Management has reviewed and approved the generated reports.

Version

Document versions:

- **Version 0:** [24/9/2025] Base document: initial general vision of the project in mind.
- **Version 1**:[12/11/2025] Functionality changes, implementation of

configuration options in profile.