**THE LITERARY COMMONS**

Figma: [Link](https://www.figma.com/design/y1zEKwltMuQQTUcwoimJHq/The-Literary-Commons?node-id=17-33&m=dev&t=ClI6pbMhAc7vzdMe-1)

GitHub: [Link](https://github.com/abhishektuteja01/the_literary_commons.git)

## Problem Statement:

## Books are considered a way to learn, educate others, draw cultural references and for entertainment. Book access, which seems like not an issue to a common man, is an issue for many communities worldwide. It can be because of affordability issues, with certain textbooks costing more than $150, or it can be because of availability of them nearby.

## Traditional libraries have sorted this problem by providing books to borrow and rent, but their inadequate inventories, strict borrowing terms, or limitations on accessibility is a concern. On the other hand, there are individual readers who possess books they no longer need, which when donated will be helpful to others.

There comes **“The Literary Commons”,** which is a community driven platform or a user-driven book-sharing system with provisions for donating, searching, borrowing and rating books easily. Instead of laying idle on shelves and gathering dust, this system helps them reach to people with low accessibility to share knowledge and educate more people.

By fostering a sense of community reading, we allow communities to share knowledge with each other, promote sustainability and create literary relationships.

## Business Requirements:

* The platform should enable users to donate, browse, borrow, and review books in a seamless and intuitive manner. The system must ensure proper cataloging, book availability tracking, and community engagement through reviews.
* Book Donations:
  + Users should be able to donate a book by providing its title, author, genre, and condition.
  + Once donated, the book should be added to the community catalog.
* Browsing and Filtering:
  + Users should be able to browse the book catalog with the ability to filter books by genre, author, and availability.
  + A search function should allow users to quickly find specific books.
* Borrowing:
  + Users should be able to mark a book as borrowed, which updates its availability status.
  + The system should clearly indicate whether a book is available or checked out.
  + Borrowed books should be listed in the user’s profile.
* Returning Books:
  + Users should also be able to mark a book as returned, making it available for others.
* Book Review and Ratings:
  + Users should be able to leave a review and rating for a book they have borrowed. Reviews should be displayed on the book’s detail page, and a rating system using a scale from one to five should be implemented.
* Storage:
  + Book data, including availability and reviews, should be stored persistently using Firestore to ensure consistency across sessions.
* User Profile:
  + Each user should have a profile displaying the books they have donated, borrowed, and reviewed.
  + The platform must be deployed for real-world access. Users should be able to interact with the system without requiring authentication, although an optional guest mode may be considered.

## Nouns and Verbs:

Nouns:

* user
* book
* title
* author
* genre
* condition
* catalog
* filters
* search function
* availability
* borrowing status
* review
* rating
* star rating system
* user profile
* persistent storage
* Firestore
* deployment
* guest mode

Verbs:

* donate
* browse
* filter
* search
* borrow
* return
* review
* rate
* store
* display
* deploy

## Summary of Modules, Attributes and Associations:

Each module represents a core component of The Literary Commons and is designed using a functional programming approach. Attributes are derived from nouns, while methods (functions) are derived from verbs and additional operations required for functionality.

1. **Book Module**

**Attributes:**

* bookId
* title
* author
* genre
* condition
* availability
* reviews
* borrowedBy

**Functions:**

* addBook(title, author, genre, condition) – Adds a new book to Firestore
* getBooks() – Retrieves all available books
* updateAvailability(bookId, status) – Updates the book’s availability (borrowed/returned)
* getBookDetails(bookId) – Fetches details for a specific book

1. **Catalog Module**

**Attributes:**

* books (list of books)
* filters
* searchQuery

**Functions:**

* filterBooks(criteria) – Filters books based on genre, author, or availability
* searchBooks(query) – Returns books matching the search term
* displayBooks() – Renders the book list in the UI

1. **Borrowing Module**

**Attributes:**

* borrowerId
* borrowedBooks (list of books borrowed by the user)
* borrowDate

**Functions:**

* borrowBook(bookId, borrowerId) – Marks a book as borrowed and updates Firestore
* returnBook(bookId, borrowerId) – Marks a book as returned and updates availability
* getUserBorrowedBooks(userId) – Retrieves books borrowed by a specific user

1. **Review & Rating Module**

**Attributes:**

* reviewId
* bookId
* userId
* reviewText
* rating

**Functions:**

* addReview(bookId, userId, reviewText, rating) – Adds a review for a book
* getReviews(bookId) – Retrieves all reviews for a specific book
* displayReviews(bookId) – Displays reviews on the book’s detail page

1. **Firestore Integration Module**

**Attributes:**

* database (reference to Firestore)

**Functions:**

* saveData(collection, data) – Stores data in Firestore
* getData(collection) – Retrieves all documents from a Firestore collection
* updateData(collection, docId, updates) – Updates a document in Firestore
* deleteData(collection, docId) – Deletes a document from Firestore

1. **User Profile Module**

**Attributes:**

* userId
* donatedBooks (list of books the user has donated)
* borrowedBooks (list of books the user has borrowed)
* reviewedBooks (list of books the user has reviewed)

**Functions:**

* getUserProfile(userId) – Fetches user profile data
* displayUserActivity(userId) – Shows donated, borrowed, and reviewed books

1. **UI Components Module**

**Functions:**

* BookCard(book) – Displays book details
* BorrowButton(bookId) – Handles borrowing action
* ReturnButton(bookId) – Handles return action
* ReviewForm(bookId) – Allows users to submit reviews
* SearchBar() – Handles user search queries
* FilterDropdown() – Enables book filtering

## Target Audience:

* **Community members** who want to share books with others. These are individuals who believe in open access to literature and want to contribute by donating books they no longer need. They are likely to be active participants in local reading initiatives and appreciate the idea of communal book-sharing.
* **Casual readers** looking for an easy way to access books without purchasing them. These users may not have the budget or desire to buy new books frequently. They prefer borrowing books on demand and appreciate a hassle-free way to explore different genres.
* **Students who need books for academic purposes** but may not want to buy them. These users rely on educational resources and textbooks but may not always have access to a library or the means to purchase every book they need. They benefit from a platform where they can borrow and return books as needed.
* **Book enthusiasts** who enjoy discovering new reads through community recommendations. These readers are passionate about literature and enjoy engaging with reviews, ratings, and book discussions. They use the platform to explore books they might not have considered otherwise and contribute by leaving reviews.
* **Individuals with extra books who want to donate them** for others to use. These users have books they have already read or no longer need and prefer donating them rather than letting them gather dust. They contribute to building a diverse catalog of books for the community.
* **Libraries or small reading groups** that want a simple way to manage shared books. These groups require an easy-to-use system to keep track of book availability and borrowing status without the complexity of a traditional library system. They rely on the platform to organize and distribute books among their members.

## Rules:

1. **Mandatory Book Details**: Users must provide the title and author when donating a book. Genre and condition are optional but recommended. Books without a title and author will not be added to the catalog.
2. **Book Availability and Borrowing:** A book marked as borrowed will no longer be available for others until it is marked as returned. Users can only borrow available books.
3. **Returning Books:** Only the user who borrowed a book can mark it as returned. Once returned, the book becomes available for others to borrow.
4. **Reviewing and Rating Books:** Users can only leave a review for a book they have borrowed. Reviews should be text-based, and ratings, if included, should be within a predefined range (e.g., 1-5).
5. **Book Removal:** Donated books cannot be removed from the system by individual users once they are borrowed. If a book is never borrowed, the donor may remove it from the catalog.
6. **Duplicate Entries:** Users cannot donate the same book multiple times. If they wish to add multiple copies, they must specify the quantity.
7. **Filtering and Search Behavior:** The filtering system will only display books matching the selected genre or author. If no books match, the catalog will indicate that no results were found.
8. **Persistent Storage Handling:** All book donations, borrowing statuses, and reviews will be stored in Firestore. The system must fetch and update book statuses dynamically to reflect real-time availability.
9. **User Profiles and Borrowing History:** Borrowing history will be tracked per user. Users can see the books they have borrowed and returned, but they cannot modify past borrowing records.
10. **Guest Access:** Users can browse the catalog without logging in. Borrowing, donating, and reviewing may require minimal user identification if needed for tracking purposes.

## Ranking Dimensions for User Personas:

1. **Reading Frequency**: Measures how often the user borrows or donates books.
2. **Community Engagement**: Represents the user’s level of interaction through reviews, discussions, and contributions to the platform.
3. **Book Ownership**: Measures how many books the user owns and whether they are more likely to donate or borrow.
4. **Tech-Savviness**: Assesses the user’s comfort level with digital tools and online platforms.

## User Personas:

1. **Emma – The Avid Book Collector**

Reading Frequency: High

Community Engagement: Low

Book Ownership: High

Tech-Savviness: Low

Emma is a retired teacher with a vast personal library. She loves collecting books but rarely lends them out or borrows new ones. She prefers physical interactions over digital platforms and needs a simple, user-friendly experience to navigate the system. She primarily donates books but does not engage much with reviews or community discussions.

User Stories:

* As a donor, I want to add books to the catalog easily so that I can contribute to the community without much effort.
* As a user who prefers offline interactions, I want a simple interface with minimal navigation so that I can donate books without confusion.

1. **Jake – The Digital Enthusiast**

Reading Frequency: High

Community Engagement: High

Book Ownership: Low

Tech-Savviness: High

Jake is a college student who reads frequently but doesn’t own many physical books. He borrows books often and actively participates in discussions, leaving reviews for every book he reads. He prefers digital platforms over physical libraries and enjoys features like filtering and search functions. He engages heavily in rating books and providing recommendations.

User Stories:

* As a reader, I want to leave reviews and ratings for books I have borrowed so that I can share my thoughts with the community.
* As a digital-first user, I want an intuitive search and filtering system so that I can quickly find books that interest me.

1. **Sophia – The Generous Donor**

Reading Frequency: Low

Community Engagement: Low

Book Ownership: High

Tech-Savviness: High

Sophia is a working professional who has accumulated a large collection of books over the years. She rarely has time to read anymore, but instead of discarding her books, she donates them in bulk. She does not borrow books and does not engage in reviews, but she finds the online platform easy to use for managing her donations.

User Stories:

* As a donor, I want to be able to donate multiple books at once so that I can quickly contribute to the community.
* As a user who does not borrow books, I want a straightforward donation process so that I can add books without unnecessary steps.

1. **Liam – The Occasional Borrower**

Reading Frequency: Low

Community Engagement: High

Book Ownership: Low

Tech-Savviness: Low

Liam is a casual reader who borrows books a few times a year but does not own many himself. He enjoys leaving reviews and discussing books with others in the community. However, he is not very comfortable with technology and prefers a simple interface to browse and borrow books.

User Stories:

* As a borrower, I want to easily mark a book as borrowed so that I can update its availability without confusion.
* As a casual reader, I want to browse books without needing an account so that I can quickly check availability.

## Use Cases:

1. **Use Case 1: Donating a Book**

Actor: User (Donor)

Goal: Add a book to the community catalog

Preconditions: The user has a book they want to donate and knows its details

Steps:

1. User navigates to the “Donate a Book” page

2. User enters book details (title, author, genre, condition)

3. User submits the form

4. The system saves the book in Firestore and updates the catalog

Postconditions: The book appears in the catalog and is available for borrowing

1. **Browsing and Filtering Books**

Actor: User (Reader)

Goal: Find books of interest

Preconditions: The catalog contains donated books

Steps:

1. User visits the catalog page

2. User can browse the available books

3. User applies filters (e.g., by genre or author)

4. System updates the displayed books based on the filters

Postconditions: The user sees a refined list of books matching their criteria

1. **Borrowing a Book**

Actor: User (Borrower)

Goal: Mark a book as borrowed and update its availability

Preconditions: The book must be available for borrowing

Steps:

1. User selects a book from the catalog

2. User clicks the “Borrow” button

3. System updates the book’s availability status in Firestore

4. The book appears in the user’s borrowed books list

Postconditions: The book is marked as borrowed and is no longer available to other users.

1. **Returning a Book**

Actor: User (Borrower)

Goal: Mark a book as returned so others can borrow it

Preconditions: The book must be in the borrowed list of the user

Steps:

1. User goes to their profile and selects a borrowed book

2. User clicks the “Return” button

3. System updates the book’s availability status in Firestore

4. The book is removed from the user’s borrowed books list

Postconditions: The book is available again for others to borrow

1. **Reviewing a Book**

Actor: User (Reader)

Goal: Leave a review and rating for a book

Preconditions: The user must have borrowed the book

Steps:

1. User selects a book they have borrowed

2. User enters a review and a rating

3. User submits the review

4. System saves the review in Firestore and updates the book’s review section

Postconditions: The review appears on the book’s detail page

## Interface Mockups:

1. **Home Page (Book Catalog)**

Description:

The home page serves as the main catalog where users can browse available books. It includes a search bar, filters for genre and author, and a list of books displaying their title, author, and availability status. Users can quickly find books they are interested in and check their availability without signing in.

User Stories Covered:

• As a digital-first user, I want an intuitive search and filtering system so that I can quickly find books that interest me.

• As a casual reader, I want to browse books without needing an account so that I can quickly check availability.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Donate a Book Page**

Description:

This page allows users to donate books to the community catalog by entering details such as title, author, genre, and condition. The interface is designed to be simple, ensuring an easy donation process without unnecessary steps.

User Stories Covered:

• As a donor, I want to add books to the catalog easily so that I can contribute to the community without much effort.

• As a user who prefers offline interactions, I want a simple interface with minimal navigation so that I can donate books without confusion.

• As a donor, I want to be able to donate multiple books at once so that I can quickly contribute to the community.

• As a user who does not borrow books, I want a straightforward donation process so that I can add books without unnecessary steps.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Book Detail Page**

Description:

This page provides detailed information about a book, including its title, author, genre, condition, and availability. If the book is available, users can mark it as borrowed. Reviews from previous readers are displayed, and users who have borrowed the book can leave their own reviews.

User Stories Covered:

• As a borrower, I want to easily mark a book as borrowed so that I can update its availability without confusion.

• As a reader, I want to leave reviews and ratings for books I have borrowed so that I can share my thoughts with the community.

A screenshot of a tablet

AI-generated content may be incorrect.

1. **User Profile Page**

Description:

The user profile page displays the books a user has borrowed and donated. It includes a “Return” button to mark borrowed books as available again and a “Review” button to submit feedback on books they have read. This eliminates the need for a separate review submission page, making it a streamlined experience.

User Stories Covered:

• As a borrower, I want to easily mark a book as borrowed so that I can update its availability without confusion.

• As a reader, I want to leave reviews and ratings for books I have borrowed so that I can share my thoughts with the community.

A screenshot of a tablet

AI-generated content may be incorrect.