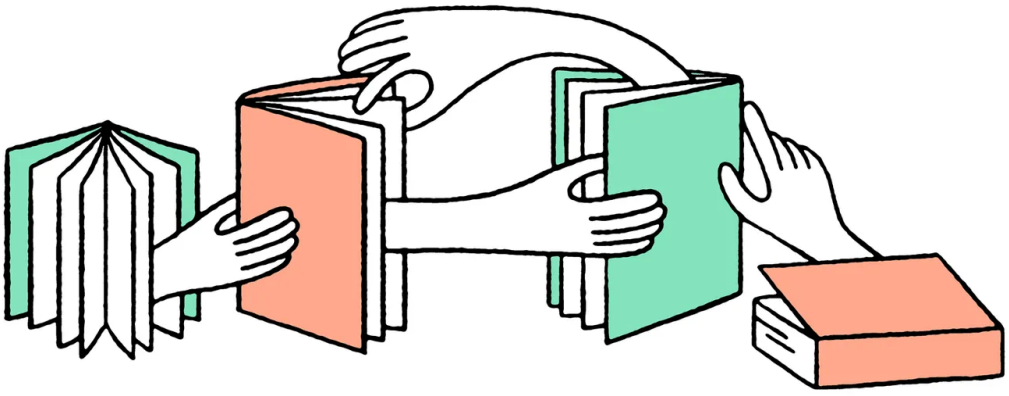
Team BiblioSwap



Devin

Kit Bazner

Lauren Inman

Table of Contents

Overview

BiblioSwap is a web application that will allow users to upload their existing library of books they are interested in trading for new books. Users will be able to browse other's libraries and initiate a swap, in which they will pay shipping in order to select a book from someone else's library. This allows the avid reader to rotate their library at a lower cost than buying new books.

Languages

BiblioSwap will be written in React and we will be using Tailwind to boost the CSS styling of the web application. We will also be using Firebase in order to manage the backend operations.C

Libraries, SDKs, APIs

- React for UI components
- Tailwind for styling
- Firebase for backend as a service (BaaS)
- Axios for HTTP requests

Initial Setup and Tests

Each library has been installed and tested on development systems to ensure compatibility.

Code Repository Organization

We will be using Git with a Trunk-Based Development strategy.

Workflow Guidelines

Frequent merging (weekly)

Pull requests and code reviews mandatory before merging

Branch naming conventions: issue numbers or features (e.g., feature/add-book)

Consultation

A meeting will be arranged with Roland Heusser to review our repo organization and workflow.

System Organization

Client/Server architecture

Frontend: React Application

Backend: Firebase BaaS

Software Architecture

(Insert figures here showing the system's software architecture)

Credit System

Earning Credits

Users can earn credits by another user purchasing their books.

Optionally, users can buy credits through in-app purchases if they don't want to list books to swap.

(1 credit for $4.99)

(3 credits for $12.99)

(5 credits for $19.99)

This way, we draw two audiences. One audience who is solely interested in finding a cheaper alternative and buys 1 credit + shipping, and the other audience who populates the platform with tons of books and self-sustains their own book swap credits by listing books.

Spending Credits

To request a book from someone else, a user must spend 1 credit.
The credit gets transferred to the account of the book owner once the swap is initiated.

Credit Rules

Credits are non-refundable and non-transferable.
One book listing equates to 1 credit in value.

Incentive Structure / Value Proposition

For Listers (Sellers): The incentive to list tons of book(s) to have a higher earning potential of credits, which can be used to acquire other books.
For Buyers: The benefit is acquiring a book at a much cheaper price (only for the cost of shipping). They also help the lister earn a credit for future transactions.

Recirculation: Because users need to list books to earn credits, and because they spend those credits to acquire other books, you create a self-sustaining ecosystem. Books are continuously added and removed from "listings libraries," keeping the platform dynamic and engaging.

Development

IDE: Visual Studio Code
OS: macOS and Windows
Deployment: AWS for production and staging
Protocols
HTTP/HTTPS for client-server communication
Custom protocol for chat (if applicable)

Possible DB Schema:

Users

- userID: Unique Identifier
- username: String
- email: String
- passwordHash: String
- location: String (Optional)
- profilePicture: URL (Optional)

Books

- bookID: Unique Identifier
- title: String
- author: String
- ISBN: String
- genre: Array of Strings
- coverImage: URL
- ownerID: Reference to User

Swaps

- swapID: Unique Identifier
- requesterID: Reference to User
- providerID: Reference to User
- bookRequestedID: Reference to Book
- bookProvidedID: Reference to Book
- status: Enum (Pending, Approved, Rejected, Completed)
- shippingCost: Number
- date: Timestamp

Chat

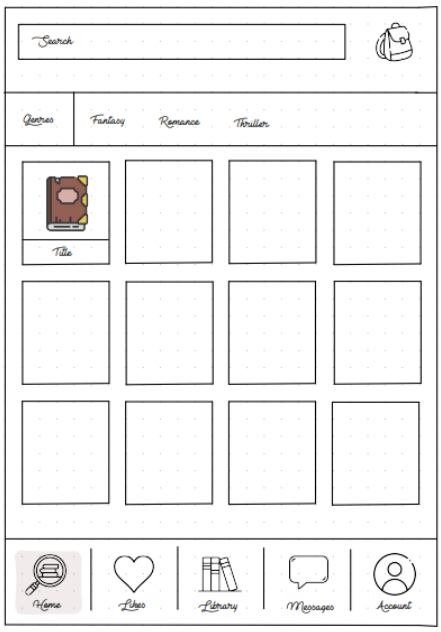
- chatID: Unique Identifier
- user1ID: Reference to User
- user2ID: Reference to User
- messages: Array of Maps
- senderID: Reference to User
- message: String
- timestamp: Timestamp

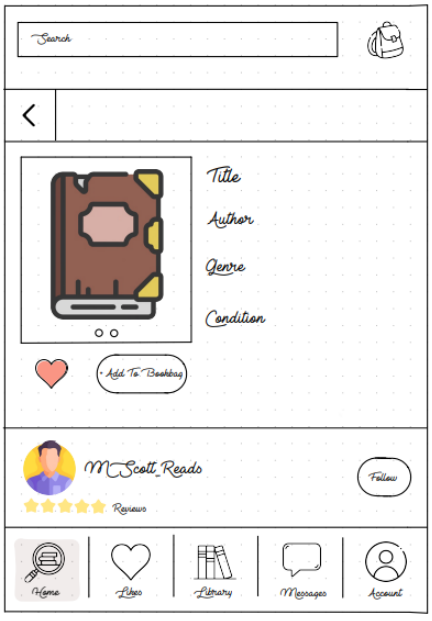
Non-Trivial Requirements

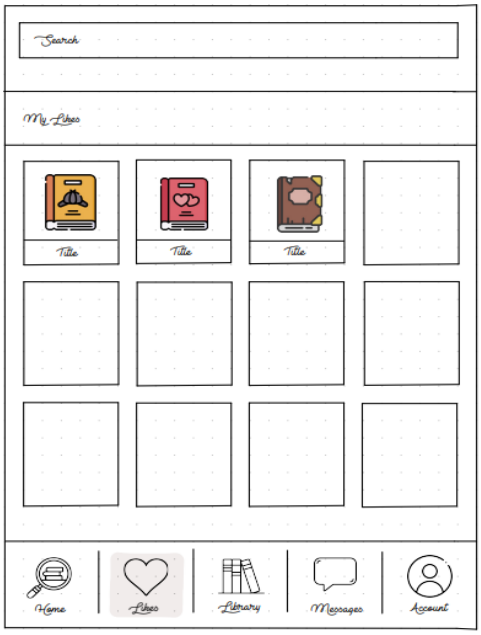
User Authentication: Possible solutions include Firebase Auth, JWT.

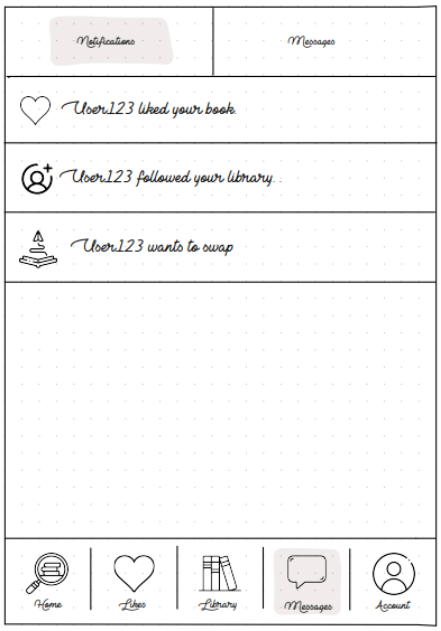
Book Recommendation Engine: Collaborative filtering, AI Incorporation

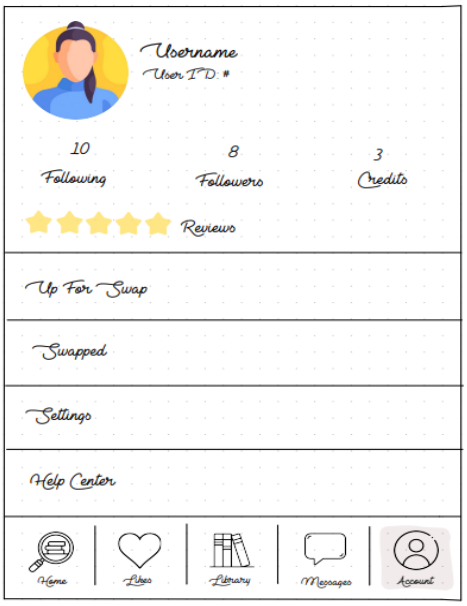
Mock-Ups











Work Policies

Daily stand-up meetings

Time-tracking using Clockify

Code reviews are mandatory

Testing

Deliverables and Milestones

Deliverables and Milestones

Sprint 1

- Create login
- Basic outline of all pages/routing

- Allow user to add a book to their library

Sprint 2

- Implement Search Functionality
- Add Book Details Page
- Implement “Library” page of favorite books

Sprint 3

- Implement Swapping Functionality / communication
- Set up Payment Estimator for Shipping

Sprint 4

- Credit system
-

Sprint 5

- Enhance UI/UX
- Optimize Performance

Sprint 6

- Final Testing
- Deployment