Implementation Report - BookTrade

Contributors: Matthew Fuchs, Sunidhi Sharma, Connor Kingsmill

Overview

BookTrade is a community-driven book sharing platform designed to facilitate peer-to-peer borrowing and lending of books within a trusted network. The platform allows users to create accounts, browse and list books, participate in discussions, and manage personalized wishlists. Admins are provided tools to moderate users and site activity.

This document outlines the features implemented, the structure of key PHP and JavaScript files, a high-level system overview, and known limitations.

Implemented Features

User Features

- **Authentication**: Users can register and log in using login.php, which validates credentials and manages session state. Sessions track whether a user is logged in and whether they are an admin.
- **Profile Management**: Users can view and update their profile information including uploading a profile picture. The form is pre-filled from MySQL, and updates are processed via php/update_profile.php.
- **Book Listings**: Logged-in users can list books for borrowing using Google Books API results. Listings are displayed in mylistings.php and stored in the borrow_listings table.
- **Book Browsing**: All users can browse available books, which are dynamically fetched and displayed with book cover images and details.
- **Wishlist**: Users can add books to a wishlist and view/remove them from their profile page.
- **Messaging**: Users can send and receive messages with other users who listed books via the messagelog table and AJAX-based messaging UI.
- **Community Discussions**: Users can create threads and reply to discussions. Discussions are stored in MySQL and displayed on discussion.php.

Admin Features

- **Dashboard**: Admins can manage users (enable/disable, edit roles) and moderate discussion threads.
- **User Editing via Modal**: Admins can edit user info using a modal popup form implemented with JavaScript and processed via php/update_user.php.

File Structure and Description

PHP Files

- index.php: Home page with conditional navigation bar based on session state.
- login.php: Handles login/signup form and redirects based on user type.
- php/login_process.php: Authenticates users, creates sessions.
- profile.php: Displays and allows updates to user profile info and wishlist.
- php/update_profile.php: Processes profile updates including image uploads.
- php/fetch_user_data.php: Retrieves user data via AJAX for dynamic profile population.
- mylistings.php: Displays books listed by the logged-in user.
- php/add_listing.php: Inserts new book entries into the borrow_listings table.
- php/remove_listing.php: Allows deletion of user's own listings.
- discussion.php: Displays discussion threads and replies.
- php/post_reply.php: Handles AJAX reply submissions.
- admin.php: Admin dashboard for managing users.
- php/update_user.php: Processes admin edits to user data.
- php/toggle_user_status.php: Enables/disables user accounts.

JavaScript Files

- admin.js: Handles admin dashboard functionality including modal population, user edit, and enable/disable toggles.
- discussion.js: Manages thread loading and asynchronous reply posting.
- messaging.js: Implements AJAX-based messaging interface between users.
- profile.js: Handles dynamic user data loading and wishlist UI updates.
- validation.js: Performs client-side validation for login, signup, and profile forms.

System Workflow (High-Level)

1. **Login & Session**: User submits credentials via login.php. Session is created and user is redirected based on role.
2. **Navigation**: Header conditionally shows login, logout, profile, or admin dashboard links based on session.
3. **Browsing Books**: Users can view book cards pulled from Google Books API or database.
4. **Posting Books**: Logged-in users can list books with metadata stored in borrow_listings.
5. **Messaging**: Each listing includes a contact button that allows messaging the lister.
6. **Profile and Wishlist**: On profile.php, user details and wishlist items are dynamically loaded and editable.
7. **Discussion System**: Users can browse or post discussions. Replies are fetched or submitted via AJAX without full reloads.
8. **Admin Dashboard**: Admins view all users, can update their info or enable/disable accounts directly from a modal interface.

Known Limitations

- **No Real-Time Messaging**: Messaging is functional but not live. Users need to refresh the message view to see new messages.

- **Basic Validation Only**: Client-side and server-side validation exist, but complex checks (e.g., password strength, rate limiting) are minimal.
- **Profile Picture Uploads**: Uploaded images are not optimized or resized, leading to potential storage issues.
- **No Email Verification**: Signup is immediate with no email verification or password reset functionality.
- **Wishlist Notifications**: Users are not automatically notified when a wished-for book becomes available.
- **Minimal Mobile Optimization**: While responsive design is implemented using Bootstrap, some pages may not be fully optimized on mobile.
- **Limited Security Hardening**: Basic protections exist, but further hardening against SQL injection, XSS, and CSRF is recommended for production use.

Conclusion

The BookTrade platform offers a foundational implementation of a functional community book-sharing system. Key features like user authentication, profile management, discussion threads, and book listing are fully integrated using PHP, JavaScript, and MySQL. The project demonstrates robust team collaboration, secure coding practices, and responsive design. Future iterations could focus on real-time features, enhanced notifications, and deeper security integration.