BookShelf

Software Design Description

Version 1.0
CMSI 4072 — Senior Project II
Kushal Jayaswal
2/20/2026

6.1 Introduction This document presents the architecture and detailed design for the BookShelf web application. BookShelf is a web based reading management platform where users can discover books, track their reading progress, write reviews, and connect with other readers. The system includes a community built book database, personal reading shelves, and basic social features such as following other users and viewing an activity feed.

6.1.1 System Objectives The objective of BookShelf is to provide users with a simple and intuitive platform for managing their reading lives. Users can search for books or add missing ones, organize their reads across personal shelves, rate and review titles they have finished, and see what their friends are reading. The application aims to make reading feel like a social and trackable activity rather than a solitary one.

6.1.2 Hardware, Software, and Human Interfaces 6.1.2.1 User Interface: The user interacts with BookShelf through a web browser. The frontend is a React single page application. Pages currently stubbed out include the landing page, registration page, login page, and a home dashboard. Navigation between pages is handled with React Router v6. Buttons and links on each page route the user to the appropriate view.

6.1.2.2 Web Browser: The application targets Google Chrome and equivalent modern browsers. A minimum display resolution of 1280x720 is required. No browser plugins or extensions are needed.

6.1.2.3 Network: Users need a broadband internet connection of at least 5 Mbps. All communication between the frontend and backend will use HTTPS.

6.1.2.4 Backend API: The backend will be a Node.js application using the Express framework, exposing a REST API. The frontend will communicate with it by sending HTTP requests and receiving JSON responses.

6.1.2.5 Database: MongoDB Atlas will serve as the database. It is a cloud hosted NoSQL document store. The backend will connect to it using the Mongoose library.

6.1.2.6 Image Hosting: Cloudinary will be used to store and serve book cover images uploaded by users. The backend will handle uploads to Cloudinary using the Cloudinary Node.js SDK.

6.1.2.7 Authentication: User passwords will be hashed using bcrypt before storage. JSON Web Tokens (JWT) will be used to maintain authenticated sessions.

6.2 Architectural Design BookShelf follows a standard three tier web architecture. The frontend (React) handles the user interface and runs in the browser. The backend (Node.js/Express) handles application logic and data access. The database (MongoDB Atlas) stores all persistent data. These three tiers communicate over HTTPS using a REST API.

6.2.1 Major Software Components Frontend: The React application is responsible for all pages the user sees and interacts with. It manages routing between pages and will eventually send requests to the backend API to load and submit data.

Backend API: The Express server receives requests from the frontend, applies business logic, and reads or writes data to MongoDB. It also handles image uploads to Cloudinary and issues JWTs for authenticated users.

Authentication Component: A dedicated part of the backend that handles user registration, login, and token verification. Every protected route checks the JWT before processing the request.

Book Component: Handles creating new book entries, searching the database by title, author, or ISBN, and returning book detail information including average rating.

Shelf Component: Manages each user's reading shelves. Handles adding, moving, and removing books from shelves and updating reading progress.

Review Component: Handles saving star ratings and text reviews, and computing the average rating for each book.

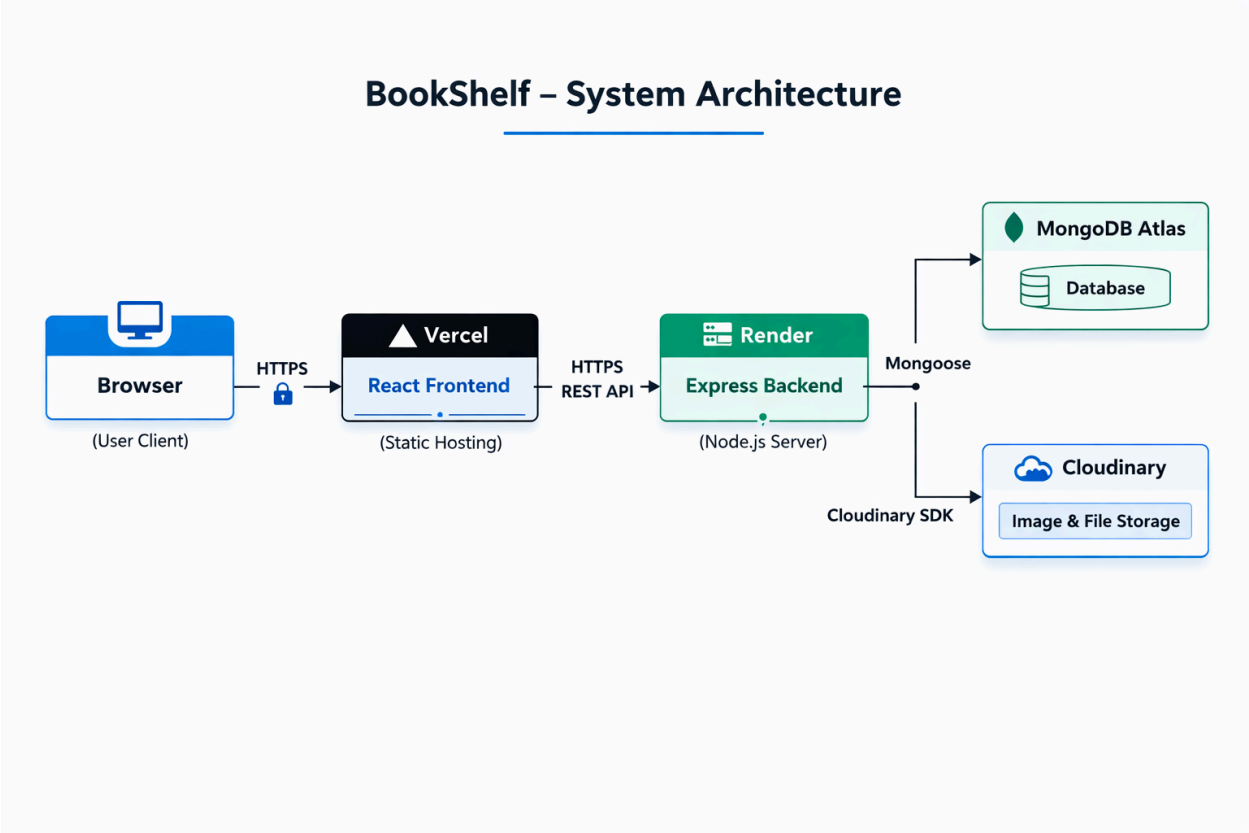
Social Component: Handles following and unfollowing other users, generating the activity feed, and serving public user profiles.

MongoDB Atlas: Stores all application data including users, books, shelves, and reviews.

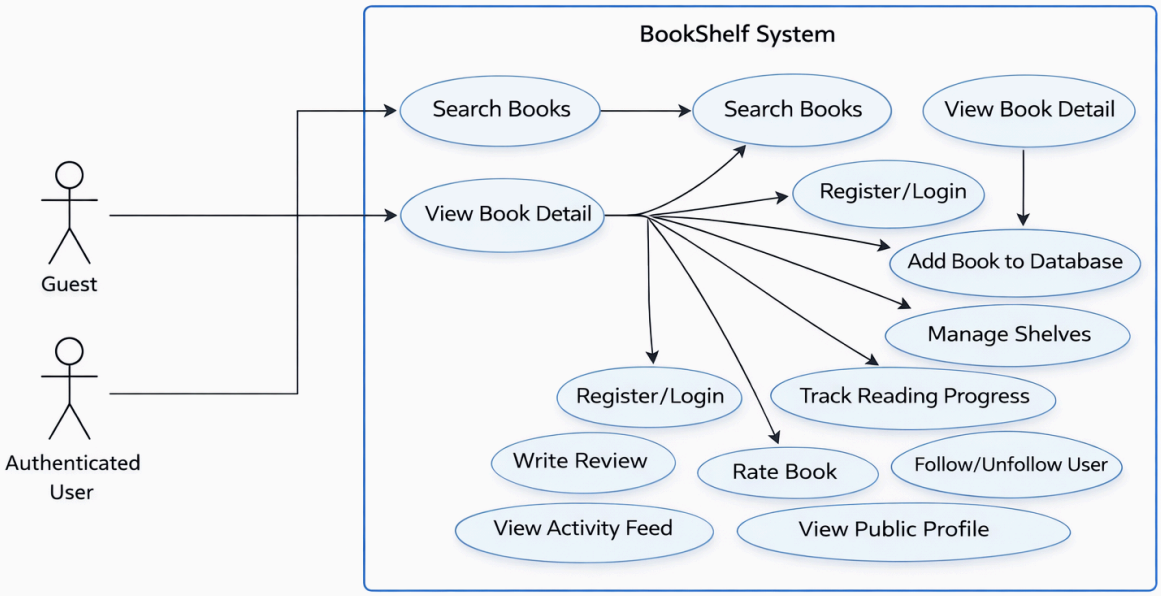
Cloudinary: External service used to store and serve book cover images.

6.2.2 Major Software Interactions The browser loads the React application from Vercel. When the user performs an action such as searching for a book or submitting a review, the React frontend sends an HTTP request to the Express backend deployed on Render. The backend validates the request, checks the JWT if the route is protected, and queries or updates MongoDB Atlas as needed. If the request involves a book cover image, the backend uploads the file to Cloudinary and stores the returned URL in the database. The backend then sends a JSON response back to the frontend, which updates the page.

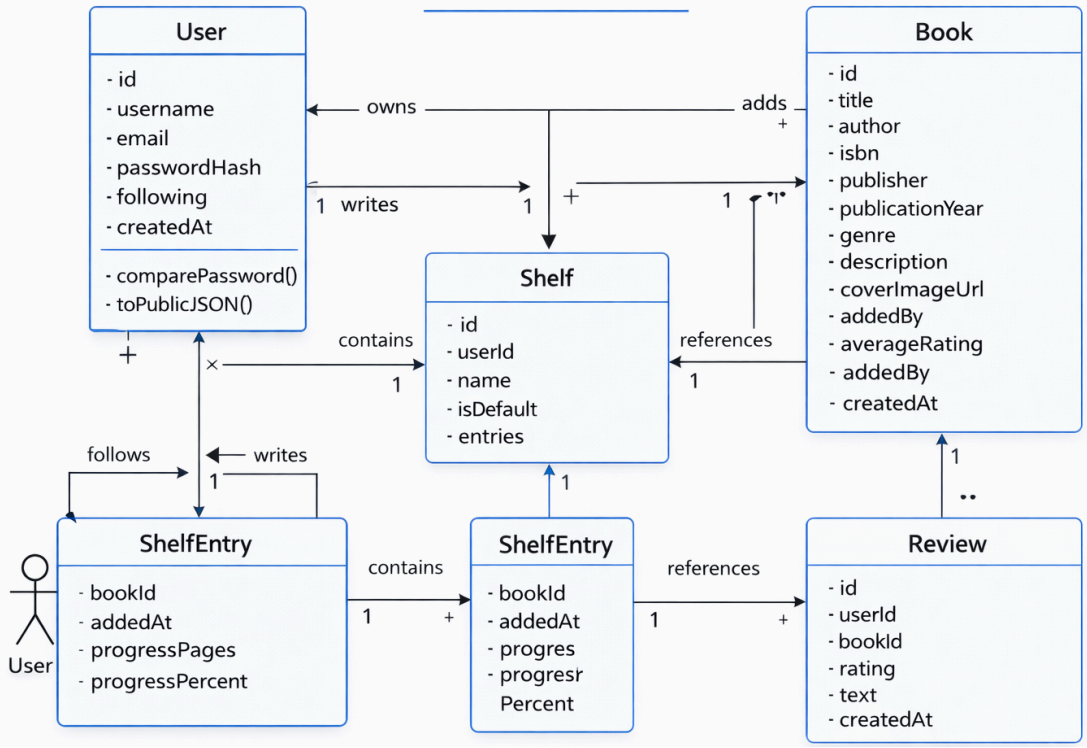
6.2.3 Architectural Design Diagrams



BookShelf System



Class Diagram



6.3. CSC and CSU Descriptions The BookShelf application is divided into three main Computer Software Components: the Frontend CSC, the Backend API CSC, and the Data Model CSC. The sections below describe the classes that make up each component.

6.3.1 Class Descriptions The following sections provide the details of all classes used in the BookShelf application.

6.3.1.1 User The User class represents a registered user of the application. It is stored in the users collection in MongoDB. It holds login credentials, display information, and a list of users that this account follows.

Fields: _id (unique identifier), username (display name), email (used for login), passwordHash (bcrypt hashed password, never returned to the client), following (list of user IDs this account follows), createdAt (date the account was created).

Methods: comparePassword(plain) — checks if a plaintext password matches the stored hash. toPublicJSON() — returns user data with the password hash removed, safe to send to the client.

6.3.1.2 Book The Book class represents a book entry in the shared database. Any authenticated user can add a book. It is stored in the books collection.

Fields: _id, title (required), author (required), isbn (optional), publisher (optional), publicationYear (optional), genre (optional), description (optional), coverImageUrl (Cloudinary URL), addedBy (ID of the user who created the entry), averageRating (calculated from all reviews), createdAt.

6.3.1.3 Shelf The Shelf class represents a named reading list that belongs to a user. Each user starts with three default shelves: Want to Read, Currently Reading, and Finished. Users can also create custom shelves. Shelves are stored in the shelves collection.

Fields: _id, userId (owner), name (shelf label), isDefault (true for the three system shelves), entries (list of ShelfEntry subdocuments).

ShelfEntry fields: bookId, addedAt, progressPages (optional, for Currently Reading), progressPercent (optional alternative to page number).

6.3.1.4 Review The Review class represents a single user's rating and optional written review for a book. Each user may only submit one review per book. Reviews are stored in the reviews collection.

Fields: _id, userId, bookId, rating (1 to 5), text (optional review body), createdAt.

6.3.1.5 AuthController Handles the /api/v1/auth routes on the backend. Responsible for user registration and login.

Methods: register(req, res) — validates email format and password length, hashes the password, saves the User, and returns a JWT. login(req, res) — looks up the user by email, checks the password, and returns a JWT on success.

6.3.1.6 BookController Handles the /api/v1/books routes. Responsible for creating book entries and searching the database.

Methods: create(req, res) — authenticated, uploads cover image to Cloudinary, saves Book document. search(req, res) — public, searches title, author, and isbn fields. getById(req, res) — public, returns a single book's details.

6.3.1.7 ShelfController Handles the /api/v1/shelves routes. All routes require authentication.

Methods: list(req, res) — returns all of the user's shelves. create(req, res) — creates a new custom shelf. addBook(req, res) — adds a book to a shelf. moveBook(req, res) — moves a book from one shelf to another. removeBook(req, res) — removes a book from a shelf. updateProgress(req, res) — updates reading progress on the Currently Reading shelf.

6.3.1.8 ReviewController Handles the /api/v1/reviews routes. Write routes require authentication.

Methods: create(req, res) — saves a review and recalculates the book's average rating. listByBook(req, res) — public, returns all reviews for a given book.

6.3.1.9 SocialController Handles the /api/v1/users routes. Write routes require authentication.

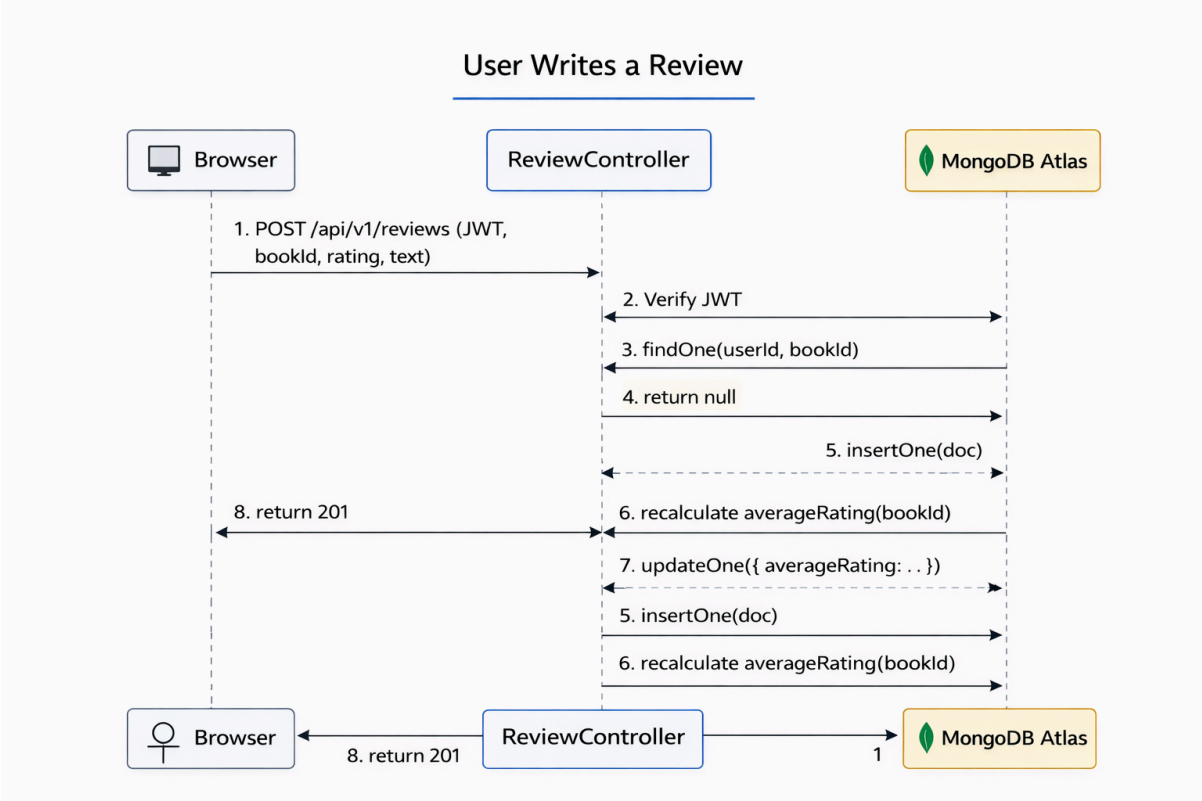
Methods: follow(req, res) — adds a user to the following list. unfollow(req, res) — removes a user from the following list. getFeed(req, res) — returns recent shelf and review activity from followed users. getProfile(req, res) — public, returns a user's public shelves and reviews.

6.3.2 Detailed Interface Descriptions The React frontend and the Express backend communicate over HTTPS using a REST API. The frontend sends JSON in the request body and receives JSON in the response. Routes that require a logged-in user expect an Authorization header containing a Bearer JWT. Errors include a JSON body with an error message field.

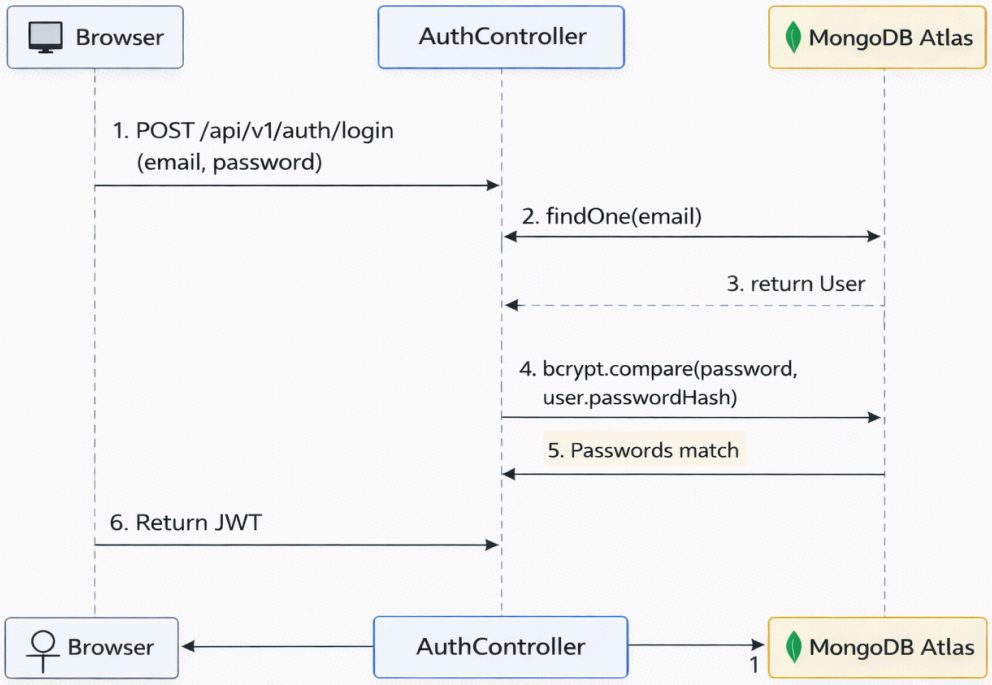
6.3.3 Detailed Data Structure Descriptions JWT Payload: Contains the user's ID, an issued-at timestamp, and an expiration timestamp set to 7 days after issue. The backend uses this token to identify the requesting user on protected routes.

Activity Feed Item: Contains the action type (shelf addition or review), the acting user's username, the book title, and a timestamp. Shelf additions also include the shelf name; reviews include the star rating. The feed is sorted newest-first and capped at 50 items.

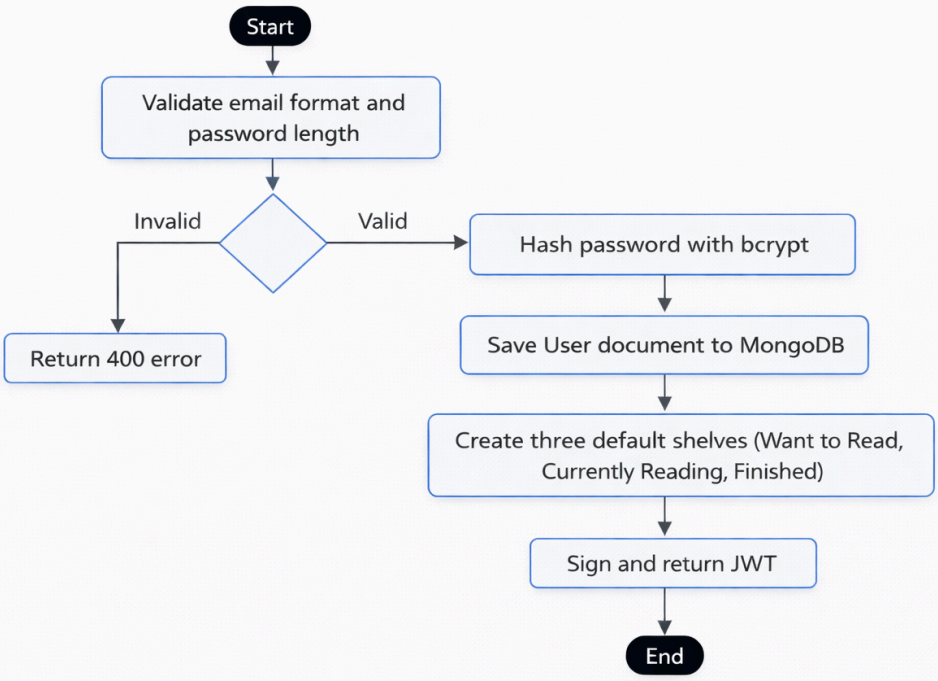
6.3.4 Detailed Design Diagrams



User Logs In

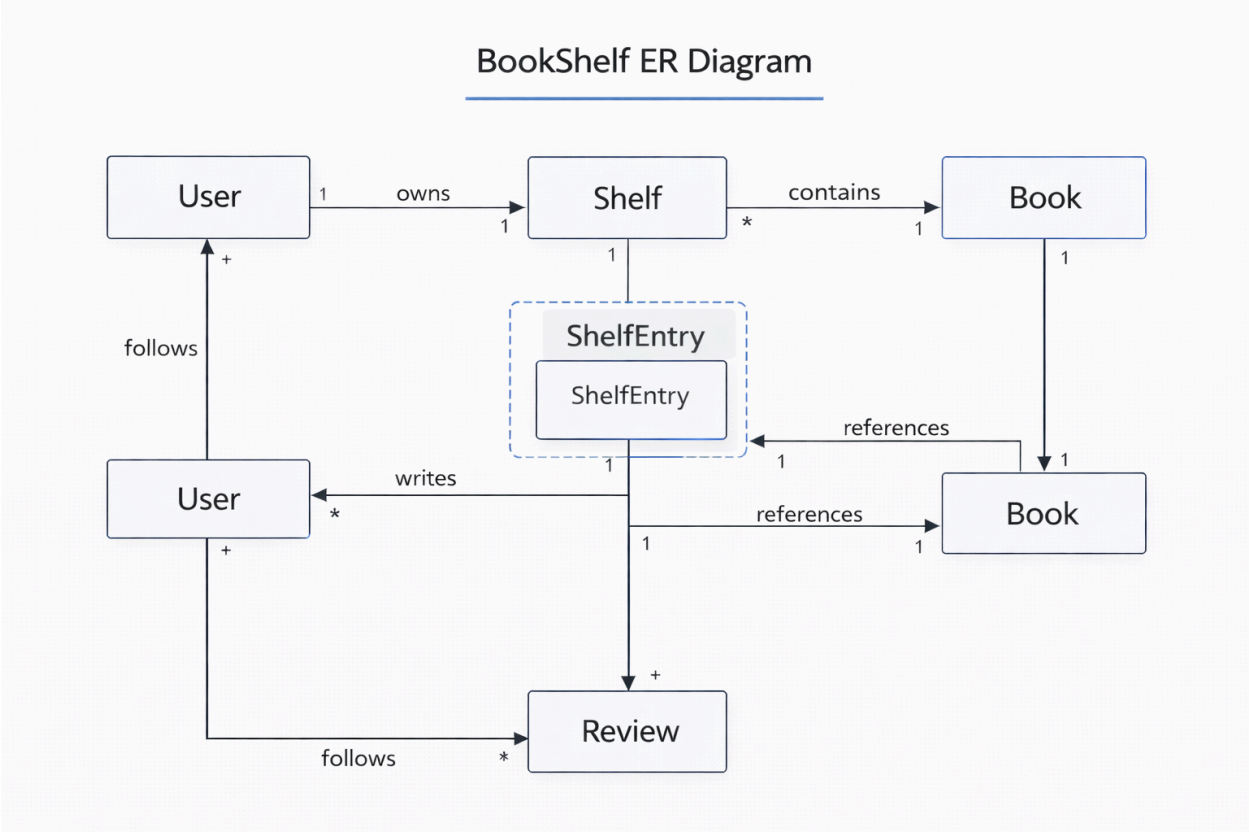


User Registration



6.4 Database Design and Description BookShelf uses MongoDB Atlas as its database. Data is organized into four collections: users, books, shelves, and reviews. The backend connects to the database through Mongoose, which provides schema validation and a convenient query interface.

6.4.1 Database Design ER Diagram



6.4.2 Database Access All database access goes through the Express backend. The frontend never contacts the database directly. Controllers use Mongoose model methods to query and update the four collections. Book search runs a case-insensitive query across the title, author, and isbn fields. Indexes are placed on users.email, users.username, and a compound index on reviews.userId and reviews.bookId to enforce the one-review-per-user-per-book rule.

6.4.3 Database Security The MongoDB Atlas cluster only accepts connections from the Render backend's IP address. The connection string is stored as a private environment variable and is never committed to the repository. The database user has read and write access only to the bookshelf database. All connections use TLS encryption. Passwords are stored only as bcrypt hashes and are never included in any API response.