Full Stack Development with MERN

Project Documentation (FSD)

BookNest: Where Stories Nestle	Team ID: LTVIP2026TMIDS24289
Team Leader: Keerthi Mekala	Member 1: Adil Mohammed
Member 2: Vasanth Metla	Member 3: Rahimtulla Shaik

1. Introduction

1.1 Project Title

BookNest: Where Stories Nestle

1.2 Team Members & Roles

Name	Role	Responsibilities
Keerthi Mekala	Team Leader / Full Stack Developer	Project planning, backend API, authentication
Adil Mohammed	Frontend Developer	React components, seller dashboard, UI design
Vasanth Metla	Backend Developer	API routes, database schema, order management
Rahimtulla Shaik	Full Stack Developer	Admin dashboard, wishlist, testing

2. Project Overview

2.1 Purpose

BookNest is a full-stack online book marketplace built using the MERN stack (MongoDB, Express.js, React.js, Node.js). The purpose of the platform is to connect book buyers and sellers on a single trusted platform, enabling users to browse, wishlist, and order books while sellers can list and manage their inventory through a dedicated dashboard. An admin panel provides oversight and control over all platform activity.

2.2 Key Features

- Secure user authentication with bcrypt password hashing for Users, Sellers, and Admins
- Book browsing with title, author, genre, description, price, and image
- Wishlist management — add and remove books
- Order placement and order history for buyers

- Seller dashboard to add, view, and delete book listings
- Seller order management — view orders placed for their books
- Admin dashboard to manage all users, sellers, and book listings
- Responsive UI using React.js, Vite, and Bootstrap
- RESTful API backend on port 4000
- MongoDB Atlas cloud database

3. Architecture

3.1 Frontend (React.js)

The frontend is a Single Page Application (SPA) built with React.js and Vite. React Router DOM handles client-side navigation between pages. Axios is used for HTTP communication with the backend API. Bootstrap provides responsive styling. The frontend runs on port 5173 during development.

Module	Pages / Components
User Module	Home, Login, Register, Wishlist, Orders, Book Detail (Uitem)
Seller Module	Seller Login, Register, Dashboard, Add Book, Seller Orders
Admin Module	Admin Login, Dashboard, View Users, View Sellers, View Books
Shared Components	Navbar, Footer, Book Card

3.2 Backend (Node.js + Express.js)

The backend is built with Node.js and Express.js, running on port 4000. It exposes a RESTful API that handles all business logic, authentication, and database interactions. Passwords are hashed with bcrypt. CORS is enabled for cross-origin requests from the React frontend. Environment variables are managed via dotenv.

3.3 Database (MongoDB)

BookNest uses MongoDB Atlas as a cloud-hosted database with Mongoose ODM for schema definition and data interaction.

Collection	Fields
users	_id, name, email, password
sellers	_id, name, email, password
admins	_id, name, email, password
items (books)	_id, title, author, genre, description, price, userId, userName, itemImage
orders	_id, userId, sellerId, bookId, title, price, buyerName, buyerAddress, createdAt

wishlists	_id, userId, bookId, title, author, price, itemImage

4. Setup Instructions

4.1 Prerequisites

- Node.js (v14 or higher)
- MongoDB Atlas account (or local MongoDB)
- npm or yarn package manager
- Git

4.2 Installation

1. Clone the repository: git clone https://github.com/Keerthi-005/BookNest1.git
2. Navigate to Backend folder: cd Book-Store/Backend
3. Install backend dependencies: npm install
4. Create a .env file in Backend with: MONGO_URL=your_mongodb_atlas_connection_string
5. Start the backend server: npm start (runs on http://localhost:4000)
6. Open a new terminal and navigate to Frontend: cd Book-Store/Frontend
7. Install frontend dependencies: npm install
8. Start the frontend: npm run dev (runs on http://localhost:5173)

5. Folder Structure

5.1 Frontend Structure

Folder / File	Description
Frontend/src/User/	All user-facing pages (Home, Login, Register, Wishlist, Orders)
Frontend/src/Seller/	Seller pages (Dashboard, Add Book, Orders)
Frontend/src/Admin/	Admin pages (Dashboard, Users, Sellers, Books)
Frontend/src/Components/	Shared reusable components (Navbar, Footer)
Frontend/src/App.jsx	Main app with routing configuration

5.2 Backend Structure

Folder / File	Description
Backend/server.js	Main entry point — all routes, middleware, and server config
Backend/db/	MongoDB connection config and Mongoose models
Backend/.env	Environment variables (MONGO_URL)

Backend/package.json	Backend dependencies and scripts

6. Running the Application

Start the backend server:

```
    cd Book-Store/Backend && npm start
```

Start the frontend development server:

```
    cd Book-Store/Frontend && npm run dev
```

Access the application at http://localhost:5173 in your browser.

7. API Documentation

7.1 User Endpoints

Method	Endpoint	Description
POST	/signup	Register a new user with name, email, password
POST	/login	Login user — returns user object {id, name, email}

7.2 Seller Endpoints

Method	Endpoint	Description
POST	/ssignup	Register a new seller
POST	/slogin	Login seller
GET	/getitem/:userId	Get all books listed by a specific seller
DELETE	/itemdelete/:id	Delete a book listing by ID
GET	/getsellerorders/:sellerId	Get all orders for a seller's books

7.3 Book Endpoints

Method	Endpoint	Description
POST	/items	Add a new book listing
GET	/item	Get all books on the platform
GET	/item/:id	Get a single book by ID
DELETE	/useritemdelete/:id	Admin: delete any book listing

7.4 Order & Wishlist Endpoints

Method	Endpoint	Description
POST	/userorder	Place an order for a book
GET	/getorders/:userId	Get all orders for a user
POST	/wishlist/add	Add a book to user's wishlist
GET	/wishlist/:userId	Get user's wishlist
POST	/wishlist/remove	Remove a book from user's wishlist

8. Authentication

BookNest uses a custom session-based authentication system without JWT tokens:

- Passwords are hashed using bcrypt before storing in MongoDB
- On successful login, the server returns user data {id, name, email} which is stored in the browser's localStorage
- Three separate login routes exist for User (/login), Seller (/slogin), and Admin (/alogin)
- Frontend checks localStorage to determine the logged-in role and controls access to role-specific pages
- On logout, localStorage is cleared and the user is redirected to the login page

9. User Interface

BookNest features a clean, responsive interface built with React.js and Bootstrap:

- Home Page — Displays all book listings with title, image, author, and price
- Book Detail Page — Shows full book details with an option to order or wishlist
- Wishlist Page — Displays all books saved by the user with remove option
- Order History Page — Lists all orders placed by the user
- Seller Dashboard — Allows sellers to add and manage their book listings
- Admin Dashboard — Provides admin with full platform management capability

10. Testing

BookNest was tested using manual testing across all features and roles:

- User Acceptance Testing (UAT) — 16 test cases covering all user, seller, and admin flows
- Functional Testing — 8 tests validating API endpoints and application logic
- Performance Testing — 6 tests measuring page load time, API response time, and multi-session stability
- Bug Tracking — Issues identified and documented with severity and resolution status

All tests passed successfully with no outstanding critical issues.

11. Known Issues

- No email notification system when a buyer places an order — seller must check dashboard manually
- No book search or filter feature — users browse all books on home page
- No payment gateway integration — orders are placed without actual payment processing
- Image upload stores base64 strings — may cause performance issues with very large images

12. Future Enhancements

- Add book search and filter by genre, author, and price range
- Integrate a payment gateway (e.g., Razorpay or Stripe) for secure transactions
- Add email notifications to sellers when orders are placed
- Implement book ratings and reviews by buyers
- Add JWT-based authentication for improved security
- Deploy the application to a cloud platform (e.g., Vercel + Render + MongoDB Atlas)
- Add an order tracking system with delivery status updates