Full Stack Development with MERN

Project Documentation format

1. Introduction

- **Project Title:** Booknest:Where Stories Nestle
- **Team Members:**M.Manasa(Team Leader)
  Md.Salma Sulthana(Team Member)
  Md.shabiha Anjum(Team Member)
  Md.Aslam(Team Member)

2. Project Overview

- **Purpose:**
  Booknest is an online platform designed to simplify book discovery, sharing, and management for readers and book collectors. It helps users browse, categorize, and track their reading preferences efficiently.
- **Features:**
  - User authentication and profiles
  - Book search and categorization
  - Book listing with reviews and ratings
  - Wishlist and currently reading tracker
  - Admin dashboard for content moderation

3. Architecture

- **Frontend (React):**
  Built using React.js with component-based architecture, React Router for navigation, and Axios for API requests. Context API or Redux is used for global state management.
- **Backend (Node.js + Express.js):**
  RESTful API structure managing user auth, book data CRUD operations, and admin functionalities. Middleware handles validation and error responses.
- **Database (MongoDB):**
  Collections include Users, Books, Categories, and Reviews. Mongoose models define schemas and relationships like user-book interactions (e.g., wishlists or reviews).

4. Setup Instructions

- **Prerequisites:**
  - Node.js & npm
  - MongoDB (local or Atlas)
  - Git
- **Installation:**

bash
CopyEdit
git clone https://github.com/yourusername/booknest.git
cd booknest
cd client && npm install
cd ../server && npm install

  - Create a .env file in /server with environment variables like MONGO_URI, JWT_SECRET.

5. Folder Structure

- **Client (React):**

  Bash
  CopyEdit
  /client
  ├── src/
  │ ├── components/
  │ ├── pages/
  │ ├── services/
  │ ├── context/
  │ └── App.js

  Components are reusable UI elements, pages represent routes like Home, Book Details, Login.

Server (Node.js):

pgsql
CopyEdit
/server
├── models/
├── routes/
├── controllers/
├── middleware/
└── server.js

Follows MVC pattern: Models define schemas, Controllers handle logic, Routes connect endpoints.

6. Running the Application

Frontend: bash
CopyEdit
cd client
npm start

Backend:

Bash
CopyEdit
cd server
npm start

7. API Documentation

- **Example Endpoints:**
  - POST /api/users/register – Register a new user
  - POST /api/users/login – Login user and return JWT
  - GET /api/books – Fetch all books
  - POST /api/books – Add a new book (admin only)
  - GET /api/books/:id – Get book details
- Each endpoint includes:
  - **Method** (GET, POST, PUT, DELETE)
  - **Request Body** (for POST/PUT)
  - **Response Example**
  - **Authentication** required (Yes/No)

8. Authentication

- **JWT (JSON Web Token)** is used for authentication.
  - Upon login, a token is issued and stored in localStorage.
  - Protected routes check for the token in headers.
  - Admin roles are verified through token payload.

9. User Interface

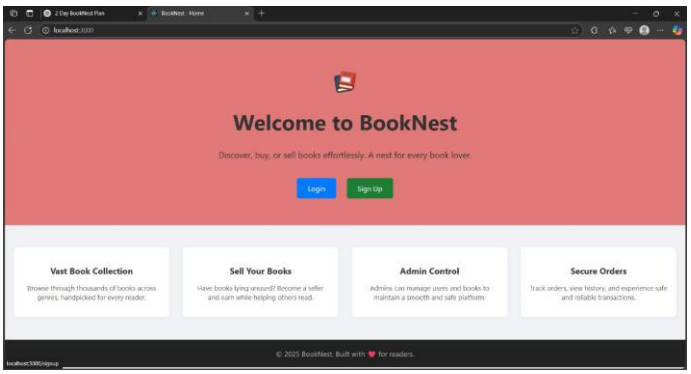
- UI includes:
    - Home page with featured books
    - Book detail modal/page
    - User dashboard with wishlist
    - Admin panel to manage listings

10. Testing

- Manual testing using Postman for API endpoints.
- Unit tests (if implemented) using Jest or Mocha.
- UI testing using tools like React Testing Library (optional).

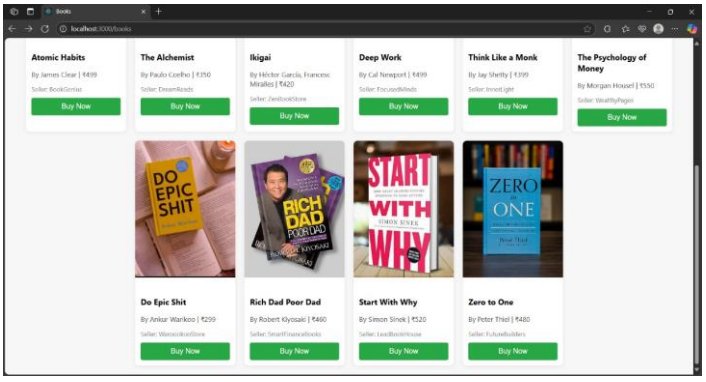
11. Screenshots or Demo

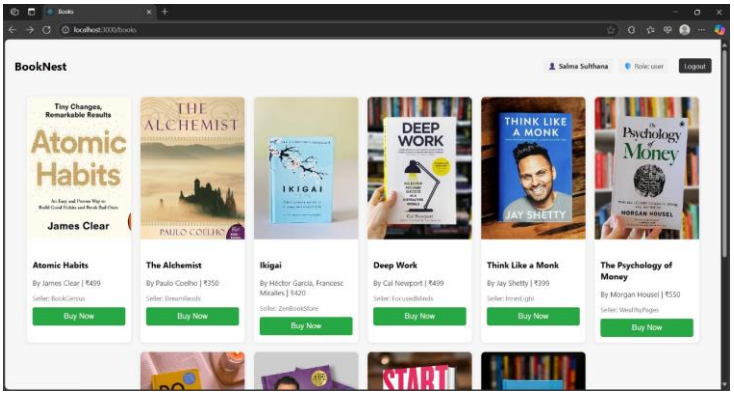
- Include actual screenshots of:
    - Landing page



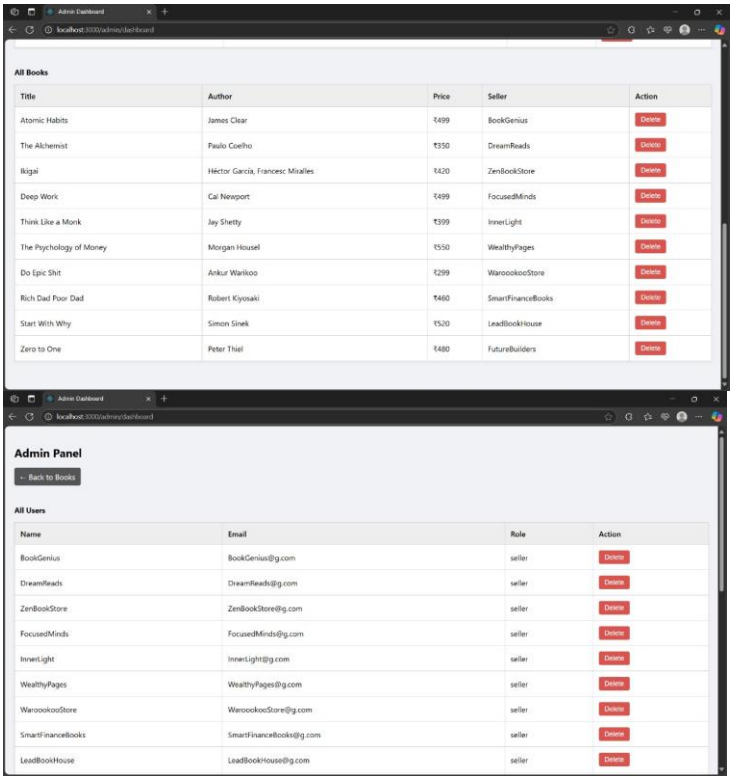
    - Login/Register

    - Book listing





Admin dashboard



12. Known Issues

- Pagination for book listings not yet implemented.
- No password reset feature yet.
- Mobile responsiveness could be improved.

13. Future Enhancements

- Add payment gateway for book purchases.
- Implement social sharing features.
- Add comment section under each book.
- Improve accessibility and mobile support.