📘 BookStore Project Documentation

Full Stack Development with MERN

---

1. Introduction

Project Title: BookStore

Team Member: Komal Tripathi

Ayushi Sharma

Shreya Sharma

Srishti Jitpure

Role & Contribution:

Team Member 1:KOMAL TRIPATHI

Backend – User & Admin Modules

Responsibilities:

- **User Module**
  - Developed the **User Model** schema using Mongoose
  - Created **controller functions** for user registration, login, wallet handling
  - Configured **user routes** for all user-related operations
- **Admin Module**
  - Developed **admin routes** for accessing the dashboard and performing book management
  - Integrated middleware for **admin authentication and authorization**
- **Implementation & Integration**
  - Connected user/admin functionalities to MongoDB
  - Handled JWT token generation and role-based access through Dotenv

---

🔹 **Team Member 2: AYUSHI SHARMA**

Backend – Books Module & Database Integration

Responsibilities:

- **Books Module**
  - Defined the **Book Model** with fields like title, author, price, and quantity
  - Built **controller functions** for CRUD operations on books
  - Configured **routes** for fetching, adding, updating, and deleting books for admin
- **Database Setup**
  - Created MongoDB collections for users, admins, and books
  - Ensured smooth connection to MongoDB via environment configuration
- **Linking and Testing**
  - Integrated routes with controllers
  - Used Postman to test all endpoints and verify backend flow

---

🔹 **Team Member 3: SHREYA SHARMA**

Frontend – Book Listing, Purchase & Wallet

Responsibilities:

- **Books Page**
  - Developed frontend to display all books using API data
  - Styled book cards using React components and CSS
- **Book Purchase Flow**
  - Implemented logic to select and purchase books
  - Added functionality to deduct book cost from user wallet
- **Wallet Management**
  - Displayed updated wallet amount post-transaction
  - Integrated with backend for syncing wallet data
- **UI Polish**
  - Ensured responsiveness and consistent user experience
  - Worked on search bar functionality and logic

🔹 **Team Member 4: SRISHTI JITPURE**

Frontend –Authentication & Admin Panel Responsibilities:

- **User & Admin Auth**
  - Designed **Login**, **Sign Up**, **Logout**, **Admin Login**, and **Admin Logout** pages

    - Handled form validations, API calls to backend auth routes
    - Managed user sessions using JWT stored in localStorage
- **Admin Dashboard**
    - Created a responsive **Admin Dashboard**
    - Integrated book management features (Add, Delete)

---

2. Project Overview

Purpose:
A full-featured online book retail platform where users can browse, search, and purchase books while admins can manage inventory and system data.

🏆 **Goals**

- Provide a responsive and intuitive user interface
- Ensure secure authentication and user management
- Maintain a clean, modular and well-structured codebase
- Design for scalability and easy maintenance

✨ **Key Features**

- **User Authentication & Authorization** – Secure login, signup, and access control
- **Admin Dashboard** – Tools for managing books, users, and analytics
- **Book CRUD Operations** – Admins can Create, Read, Update, and Delete books
- **Environment-based Configuration** – Uses .env files for flexible deployment
- **MongoDB Schema Validation** – Ensures consistent and validated data using Mongoose schemas

---

3. Architecture

Frontend (React)

- ✅ **React Functional Components:**
  These are simpler, modern React components defined as JavaScript functions — easy to write, read, and test.
- 🔁 **React Hooks:**
  Hooks like useState and useEffect let you add state and side effects to functional components without using classes.
- 🧭 **React Router:**
  Handles navigation between pages (like /home, /login, /book/:id) without refreshing the whole page.

- 🌐 **Axios:**
  A promise-based HTTP client used to make API requests (like GET /books or POST /login) to the backend.
- 📦 **Context API:**
  Allows you to share state (like user info) globally across your app without prop-drilling.

⚙️ **Backend (Node.js & Express.js)**

- 🔗 **RESTful APIs:**
  Endpoints like GET /books or POST /login follow REST conventions to handle CRUD operations.
- 📂 **Route Controllers:**
  Separate logic that handles what each route should do (e.g., fetching books, registering users).
- 🌍 **CORS (Cross-Origin Resource Sharing):**
  Middleware that allows your frontend (on one domain/port) to communicate with your backend (on another).
- 📁 **Node_modules/bytes:**
  node_modules contains installed packages. bytes is a utility to parse and format byte values (often used in uploads or limits).

🗄️ **Database (MongoDB + Mongoose)**

- 📚 **Collections: Users, Books:**
  Collections are like tables in SQL. You have one for users and one for books, storing their respective data.
- ✅ **Data Validation & Indexing:**
  Mongoose enforces rules (like required fields) and creates indexes to speed up searching/filtering.
- 📦 **Object Modeling with Mongoose:**
  Mongoose lets you define schemas and models, which act like blueprints for your MongoDB documents.

---

4. Setup Instructions

Prerequisites:

To develop, run, and deploy the application effectively, the following software and tools were required throughout the project

- **Node.js (v16 or later):** For running the backend server and handling server-side operations.
- **npm:** For managing project dependencies such as mongoose, cors, b, moment.
- **MongoDB (v5 or later):** Used as the NoSQL database to store user, expense, and category data, either locally or via MongoDB Atlas
- **Mongoose:** Simplified database interactions through schema definitions and validation.
- **Git:** Enabled version control and team collaboration throughout the project.
- **Code Editor (e.g., VS Code):** Used for developing and managing the codebase.
- **Web Browser**: Required for testing and interacting with the frontend.
- **Postman / Thunder Client (optional):** Helpful for testing backend API endpoints.

Installation:

📥 **Step 1: Clone the Repository**

1. Go to the GitHub repo: https://github.com/KomalTripathi/SmartBridge.git
2. Click on the green **Code** button and copy the repository URL or download the ZIP file.

If you're using **Git**:

📁 **Step 2: Create a Project Folder**

Open **Command Prompt (CMD)** and run the following commands:

1. cd Desktop
2. KomalTripathi SmartBridgeClone
3. cd MernBookstoreClone
4. git clone https://github.com/KomalTripathi/SmartBridge.git
5. cd MernBookStore
6. code .

```
cd Desktop
KomalTripathi SmartbridgeClone
cd SmartbridgeClone
git clone https://github.com/KomalTripathi/SmartBridge.git
cd SmartBridge
code .
```

This will open the project in **Visual Studio Code**.

▶️ Step 3: Run the Application

Open the **integrated terminal** in VS Code and split it into two parts:

🖥️ Frontend (React)

cd MernBookStore/frontend

npm install

npm run dev

```
cd MernBookStore/frontend
npm install
npm run dev
```

- This will start the frontend server.
- Click on the link shown in the terminal (usually http://localhost:5173 or similar).

⚙️ Backend (Node.js + Express)

In the second terminal:

cd MernBookStore/backend

npm install

npm start

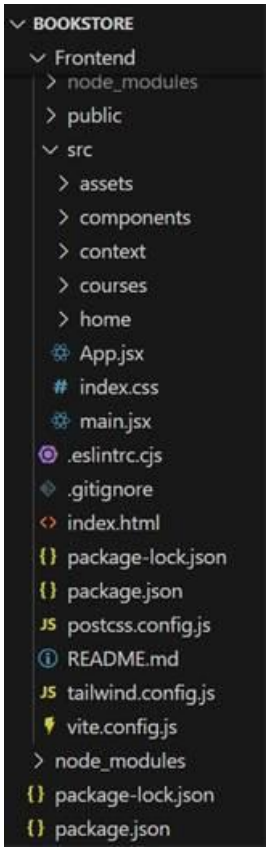
```
cd MernBookStore/backend
npm install
npm start
```

This starts the backend server (usually on http://localhost:5000).

5. Folder Structure

Client (React Frontend)

- The frontend is organized with reusable **functional components** (e.g., Navbar, Home, BookDetails).
- **React Router** manages page navigation (/login, /signup, /books, etc.).
- **Axios** is used to make API calls to the backend.
- **Context API** handles shared state like user authentication and wallet balance.
- Static assets (like images and CSS) are stored in the public and assets folders.



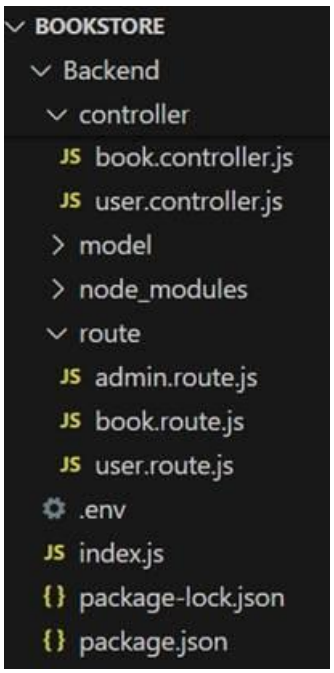
📂 **Backend:**

1. Backend uses **Express.js** with a modular structure:

    1. routes/ handles API endpoints for users and books.
    2. controllers/ contains logic for handling route requests (e.g., login, signup, book search).
    3. models/ defines Mongoose schemas for User and Book.

    d. middleware/ (if added) would include auth-checking logic

2. MongoDB is connected using **Mongoose**, and .env is used for environment variables (like DB connection strings and JWT secrets).



---

6. Running the Application

Provide commands to start the frontend and backend servers locally.

▶️ **Start Backend:**

```
cd backend

npm install

npm start
```

▶️ **Start Frontend:**

```
cd frontend

npm install

npm run dev
```

- Frontend runs on: http://localhost:4001
- Backend runs on: http://localhost:27017

---

7. API Documentation

🔐 Authentication Routes:

import express from "express";

import { signup, login, purchaseBook, getWalletBalance } from "../controller/user.controller.js";

const router = express.Router();

router.post("/signup", signup);

router.post("/login", login);

router.post("/purchase", purchaseBook);

router.get("/wallet/:id", getWalletBalance);

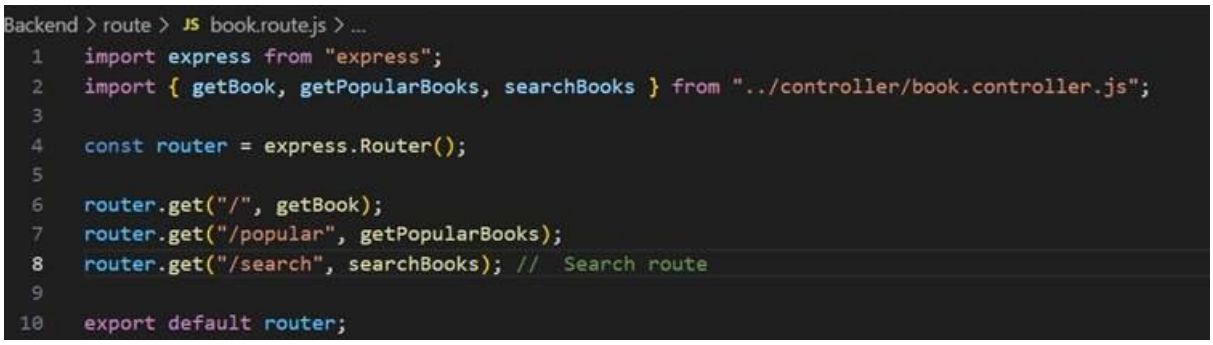
export default router;

These are the key user-related endpoints defined:

- POST /signup → Registers a new user.
- POST /login → Logs in an existing user and returns a JWT.
- POST /purchase → Authenticated route to purchase a book.
- GET /wallet/:id → Authenticated route to get user's wallet balance.

📚 Book Routes:

```
import express from "express";

import { getBook, getPopularBooks, searchBooks } from "../controller/book.controller.js";

const router = express.Router();

router.get("/", getBook);

router.get("/popular", getPopularBooks);

router.get("/search", searchBooks); //  Search route

export default router;
```



These are accessible without login:

- GET / → Fetch all books.
- GET /popular → Get trending books.
- GET /search → Search books by keyword.

---

8. Authentication & Authorization

In the BookStore project, authentication is handled using JWT tokens that are generated upon login and stored in the browser's localStorage. Passwords are securely hashed using bcrypt before being saved. Authorization is role-based, distinguishing users from admins where needed. Protected routes (e.g., purchases, wallet access) check for valid JWT tokens before processing. There's no session management—authentication relies entirely on token validation.

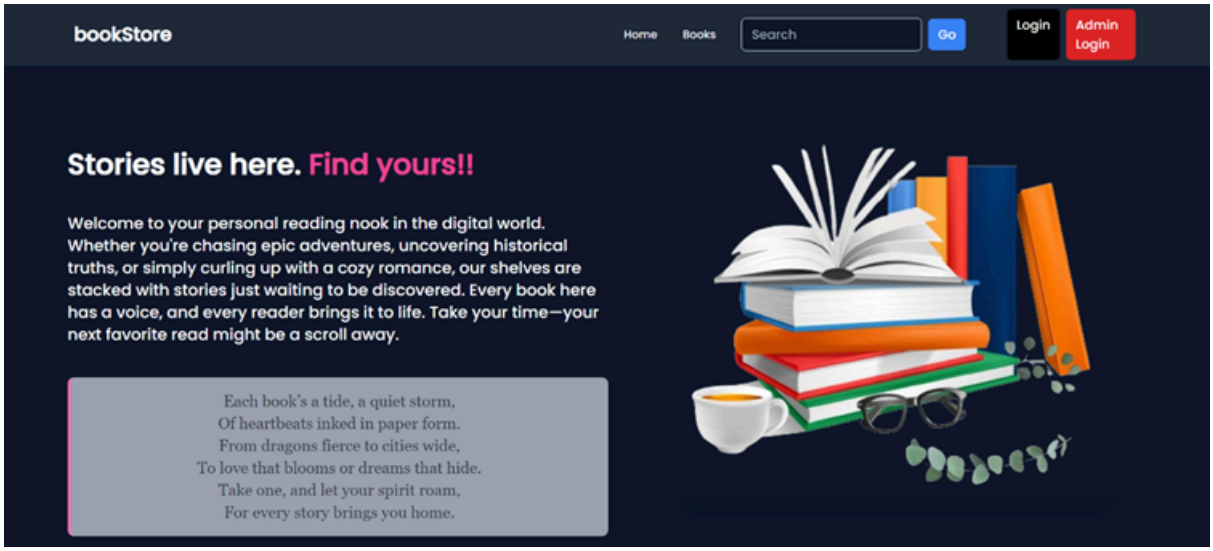
1. **User Authentication** is handled using JWT (JSON Web Tokens) upon successful login or signup.

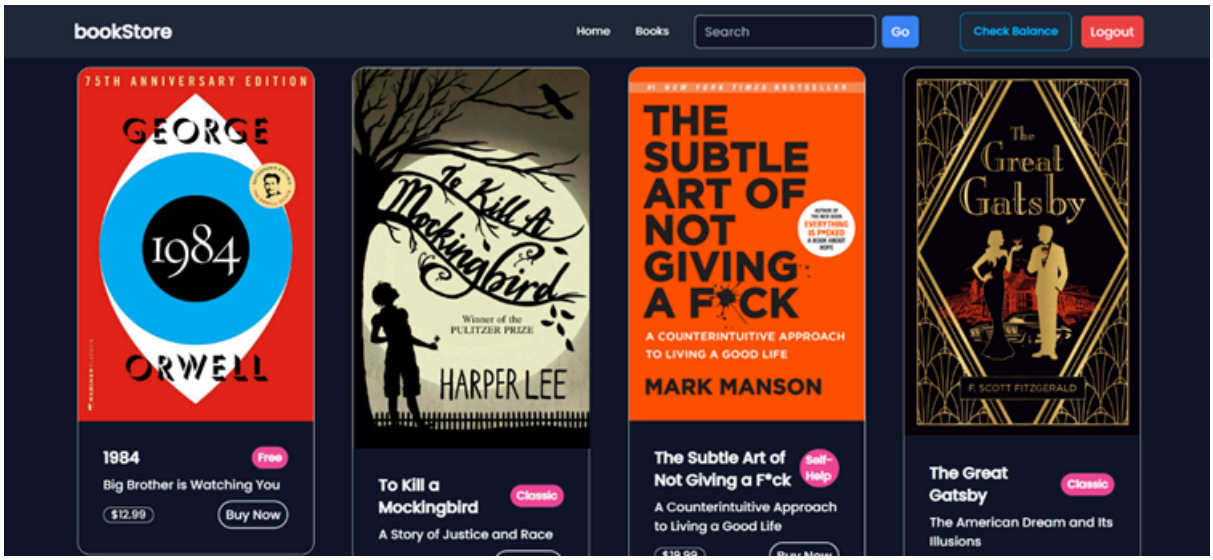
2. **Tokens** are generated in the backend and stored in the browser's localStorage on the frontend to maintain user sessions.

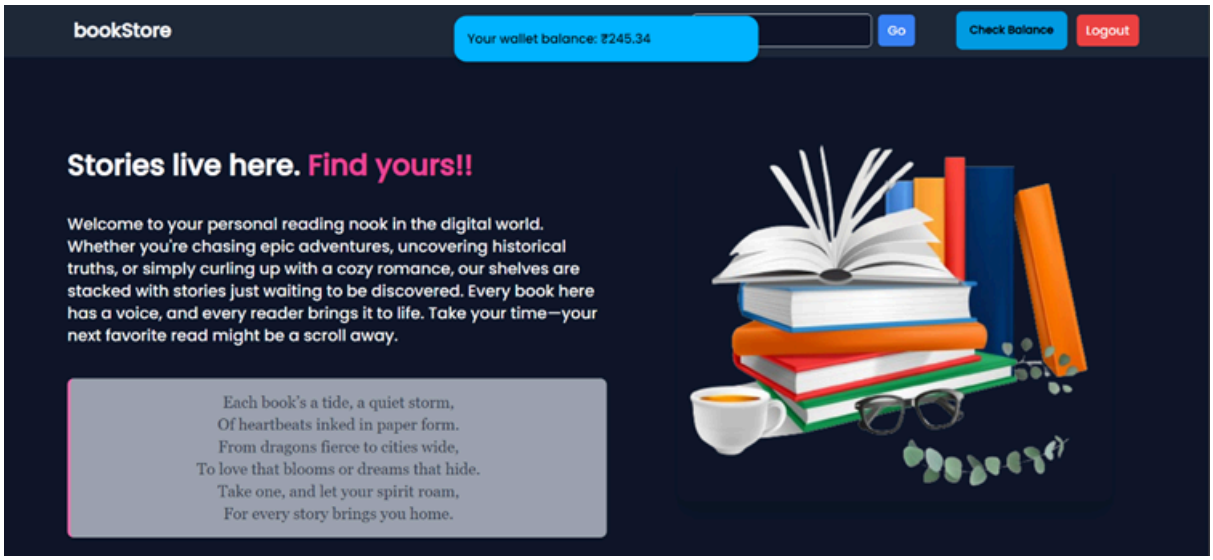
3. **Passwords** are securely hashed using bcrypt before being saved in the MongoDB database.
4. **Routes** like /purchase and /wallet/:id are protected, meaning they require a valid token to access user-specific data or actions.
5. **Role-based authorization** (e.g., admin-only features) can be managed by checking the user's role stored in the token payload (though admin-specific routes are not yet implemented in the current repo).
6. **Dotenv-based Authentication:** Environment variables securely store sensitive credentials like JWT secrets.
7. **Tokens Stored in localStorage:** JWT tokens are saved in the browser's localStorage for persistent user sessions.
8. **Role-based Access for Admin:** Admin-specific features are protected based on user roles.
9. **Passwords Hashed with bcrypt:** User passwords are encrypted using bcrypt before storing in the database.
10. **Protected Routes using CORS:** Cross-Origin Resource Sharing ensures secure API communication between frontend and backend.

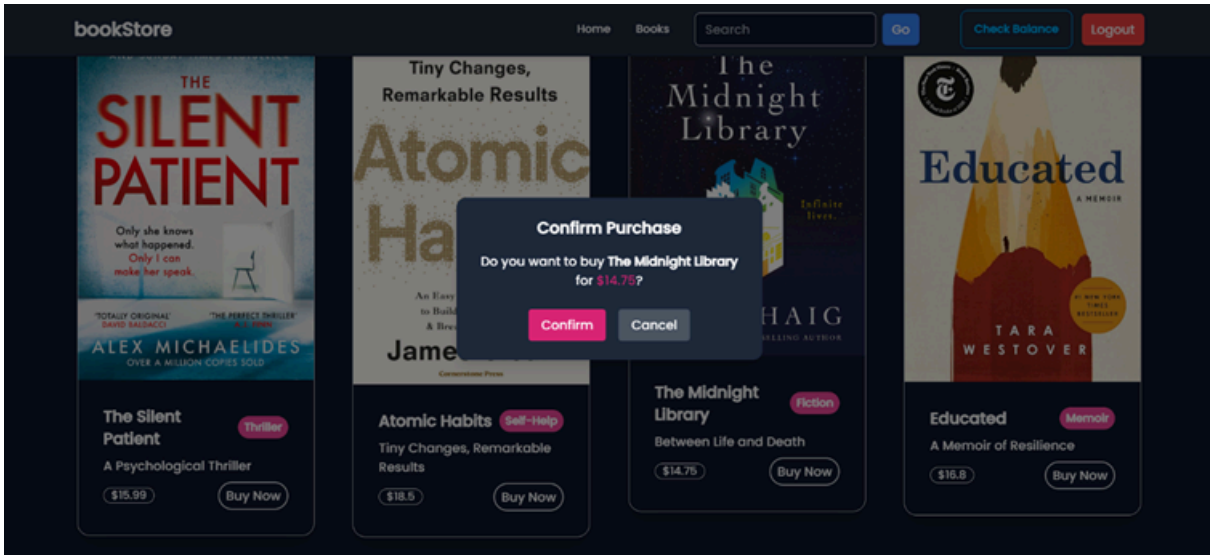
---

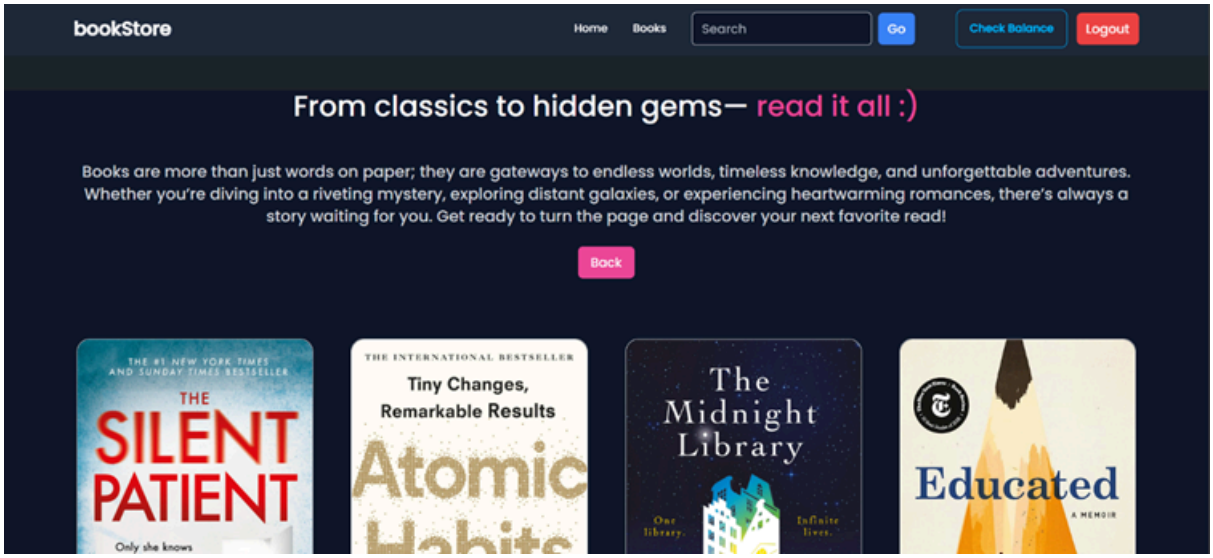
9. User Interface






bookStore
Your wallet balance: ₹245.34
Go
Check Balance
Logout
Stories live here. Find yours!!
Welcome to your personal reading nook in the digital world. Whether you're chasing epic adventures, uncovering historical truths, or simply curling up with a cozy romance, our shelves are stacked with stories just waiting to be discovered. Every book here has a voice, and every reader brings it to life. Take your time—your next favorite read might be a scroll away.
Each book's a tide, a quiet storm,
Of heartbeats inked in paper form.
From dragons fierce to cities wide,
To love that blooms or dreams that hide.
Take one, and let your spirit roam,
For every story brings you home.


bookStore
Home
Books
Search
Go
Check Balance
Logout
THE SILENT PATIENT
Only she knows what happened. Only I can make her speak.
ALEX MICHAELIDES
OVER A MILLION COPIES SOLD
The Silent Patient
Thriller
A Psychological Thriller
$15.99
Buy Now
Tiny Changes, Remarkable Results
Atomic Habits
Self-Help
Tiny Changes, Remarkable Results
$18.5
Buy Now
Confirm Purchase
Do you want to buy The Midnight Library for $14.75?
Confirm
Cancel
The Midnight Library
Fiction
Between Life and Death
$14.75
Buy Now
Educated
A MEMOIR
TARA WESTOVER
Educated
Memoir
A Memoir of Resilience
$16.8
Buy Now


bookStore
Home
Books
Search
Go
Check Balance
Logout
From classics to hidden gems— read it all :)
Books are more than just words on paper; they are gateways to endless worlds, timeless knowledge, and unforgettable adventures. Whether you're diving into a riveting mystery, exploring distant galaxies, or experiencing heartwarming romances, there's always a story waiting for you. Get ready to turn the page and discover your next favorite read!
Back
THE SILENT PATIENT
Tiny Changes, Remarkable Results
Atomic Habits
The Midnight Library
Educated

Admin Dashboard
Hello, Admin
Logout
Books
Add Book
Users
Book Popularity
The Silent Patient
Price: ₹15.99
Delete
Atomic Habits
Price: ₹18.5
Delete
The Midnight Library
Price: ₹14.75
Delete
Educated
Price: ₹16.8
Delete
1984
Price: ₹12.99
Delete
To Kill a Mockingbird

10. Testing

Feature / Functionality	Description
User Authentication	Login, Registration, dotenv jwt key Authentication
Dashboard	Access after login, view balance, and available books
Search Functionality	Real-time book search
Book Purchase	Buying books using initial $400 credit
Admin Features	View user sign ups, book popularity stats, add new books

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
TC-001	User Registration	1. Open Register Page 2. Enter Email, Password, Confirm Password 3. Click Submit	User gets registered and redirected to dashboard	User gets registered and redirected to dashboard	[✔]
TC-002	User Login	1. Open Login Page 2. Enter valid	Redirected to dashboard with token saved	Redirected to dashboard with token saved	[✔]

		credentials 3. Click Login			
TC-003	Book Search	1. Navigate to Search Bar 2. Type Book Name	Matching books appear dynamically	Matching books appear dynamically	[✔]
TC-004	Purchase Book	1. Select Book 2. Click Buy 3. Confirm Purchase	$ deducted, book added to profile	$ deducted	[✖]
TC-005	Admin Adds Book	1. Login as Admin 2. Go to Admin Panel 3. Fill book form, submit	New book is visible in catalog	New book is visible in catalog	[✔]

---

11. Screenshots or Demo

demo video:

https://drive.google.com/drive/folders/1rNeJxNFgEf1HynxFmUcf7jo_f0Pq_MuS?usp=sharig

---

12. Known Issues

No Pagination: All data loads at once, making it harder to manage large datasets.

No Password Reset: Users can't recover or change their password if they forget it.

No Email Verification: Users can register without confirming their email address.

---

13. Future Enhancements

1. **Full Order Workflow:** Manages the complete process from book selection to final purchase confirmation.
2. **Payments & Checkout:** Enables secure online payments through integrated gateways.
3. **Book Reviews and Ratings:** Allows users to rate and review books they've purchased.
4. **Search and Filtering:** Helps users quickly find books using keywords and filters.
5. **Admin Analytics:** Provides visual insights into sales, users, and performance for admins.
6. **Multi-language Support (i18n):** Supports multiple languages for a globally accessible user interface.

+