📁 Project Structure and coding

BookNest/

├── client/ # React frontend

│ ├── public/

│ │ └── index.html

│ ├── src/

│ │ ├── components/

│ │ │ ├── BookCard.js

│ │ │ ├── BookList.js

│ │ │ ├── Cart.js

│ │ │ ├── Header.js

│ │ │ └── LoginForm.js

│ │ ├── pages/

│ │ │ ├── HomePage.js

│ │ │ ├── BookPage.js

│ │ │ └── AdminPage.js

│ │ ├── api/

│ │ │ └── api.js

│ │ ├── App.js

│ │ └── index.js

│ └── package.json

├── server/ # Express + Node backend

│ ├── controllers/

│ │ ├── authController.js

│ │ └── bookController.js

│ ├── middleware/

│ │ └── authMiddleware.js

│ ├── models/

│ │ ├── Book.js

│ │ └── User.js

│ ├── routes/

│ │ ├── authRoutes.js

│ │ └── bookRoutes.js

│ ├── config/

│ │ └── db.js

│ ├── server.js

│ └── package.json

├── .gitignore

└── README.md

🛠️ Key Files

Backend – server/server.js

import express from 'express';

import mongoose from 'mongoose';

import dotenv from 'dotenv';

import authRoutes from './routes/authRoutes';

import bookRoutes from './routes/bookRoutes';

import { connectDB } from './config/db';

dotenv.config();

connectDB();

const app = express();

app.use(express.json());

app.use('/api/auth', authRoutes);

app.use('/api/books', bookRoutes);

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

MongoDB Connection – server/config/db.js

import mongoose from 'mongoose';

export const connectDB = async () => {

try {

await mongoose.connect(process.env.MONGO\_URI);

console.log('MongoDB connected');

} catch (err) {

console.error(err);

process.exit(1);

}

};

User Model – server/models/User.js

import mongoose from 'mongoose';

const userSchema = new mongoose.Schema({

name: String,

email: { type: String, unique: true },

password: String,

isAdmin: { type: Boolean, default: false },

});

export default mongoose.model('User', userSchema);

Book Model – server/models/Book.js

import mongoose from 'mongoose';

const reviewSchema = new mongoose.Schema({

user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

stars: Number,

comment: String,

}, { timestamps: true });

const bookSchema = new mongoose.Schema({

title: String,

author: String,

genre: String,

price: Number,

rating: { type: Number, default: 0 },

reviews: [reviewSchema],

}, { timestamps: true });

export default mongoose.model('Book', bookSchema);

Book Routes & Controller (CRUD, reviews)

routes/bookRoutes.js

import express from 'express';

import { getBooks, getBookById, addReview } from '../controllers/bookController';

import { protect } from '../middleware/authMiddleware';

const router = express.Router();

router.route('/').get(getBooks);

router.route('/:id').get(getBookById).post(protect, addReview);

export default router;

controllers/bookController.js

import Book from '../models/Book.js';

export const getBooks = async (req, res) => {

const books = await Book.find({});

res.json(books);

};

export const getBookById = async (req, res) => {

const book = await Book.findById(req.params.id);

if (book) res.json(book);

else res.status(404).json({ message: 'Book not found' });

};

export const addReview = async (req, res) => {

const { stars, comment } = req.body;

const book = await Book.findById(req.params.id);

if (book) {

const review = { user: req.user.\_id, stars, comment };

book.reviews.push(review);

book.rating = book.reviews.reduce((sum, r) => sum + r.stars, 0) / book.reviews.length;

await book.save();

res.status(201).json({ message: 'Review added' });

} else res.status(404).json({ message: 'Book not found' });

};

Authentication Routes

routes/authRoutes.js, controllers/authController.js, using bcrypt and JWT - for login/register functionality with token issuance and protect middleware.

💻 Frontend Highlights

api/api.js using axios:

import axios from 'axios';

export default axios.create({ baseURL: '/api' });

React Context or Redux for auth and cart state management.

Pages/components include listing, book details, cart checkout flow (mock), admin add/edit book forms.

🪜 Setup Instructions

1. Create .env in server:

MONGO\_URI=your-mongo-connection-url

JWT\_SECRET=your\_jwt\_secret

2. In server/ run:

npm install

npm run dev

3. In client/ run:

npm install

npm start

✅ What’s Included

Fully structured client and server code.

Models: User, Book with rating/review support.

Routes: auth, book browsing, review submission.

JWT authentication and authorization.

Frontend core components and API integration skeleton