

# PROJECT ON BC

PROJECT ON  
BOOK STORE USING MONGODB

BY TEAM MEMBERS NAME :

NAME :

Venkatanrayana G.V-311421205102  
Vigneshwaran S-311421205103  
Sam Rohid F-311421205075  
Vishwa L-311421205106

Venkatanrayana G.V-311421205102  
Vigneshwaran S-311421205103  
Sam Rohid F-311421205075  
Vishwa L-311421205106

## **1. Introduction :**

In this project, we develop a Book Store application using MongoDB as the database

backend. MongoDB is a NoSQL database known for its scalability and flexibility in

handling large volumes of unstructured data. This application provides a platform for

users to view and purchase books, along with an administrative interface to manage

books and orders. By using MongoDB, we can easily handle diverse book data (e.g.,

title, author, price) and user activity, all in a schemaless format

## 2. System Requirements

To run the Book Store application successfully, the following system requirements must be met

Hardware :

Processor: 2 GHz or higher

RAM: 4 GB minimum (8 GB recommended)

Storage: 1 GB of free space for application and database storage

Software:

Operating System: Windows, macOS, or Linux

MongoDB: Version 4.x or higher

Node.js: Version 14.x or higher

npm: Version 6.x or higher

Web Browser: Latest version of Chrome, Firefox, or Safari

Development Tools:

IDE/Editor: Visual Studio Code or any preferred code editor

### 3. Prerequisite :

Before you start implementing the Book Store application, ensure the following prerequisites:

**MongoDB Setup:** Ensure that MongoDB is installed and running on your local machine or on a cloud service (e.g., MongoDB Atlas).

**Node.js and npm:** Install Node.js and npm (node package manager) for setting up the application server.

**Version Control:** Git should be installed for version control and collaboration.

**Knowledge of JavaScript/Node.js:** The application will use JavaScript with Node.js for the backend.

## 4. Architecture :

The Book Store application follows a ClientServer architecture with a RESTful API connecting the frontend (user interface) to the backend (server and database).

Key components of the architecture:

Frontend (Clientside):

A webbased interface for users to browse, purchase, and review books.

Developed using HTML, CSS, and JavaScript.

Backend (Serverside):

Node.js server hosting the RESTful API.

Express.js to manage routes and API endpoints.


MongoDB to store and retrieve bookrelated data, user profiles, and orders.

Database (MongoDB):

MongoDB collections for books, users, and orders.

Data stored in JSON like documents (BSON format).

Flow:

1. User Interaction: The user interacts with the frontend through a web browser.
  2. API Requests: The frontend sends HTTP requests (GET, POST, PUT, DELETE) to the backend.
  3. Database Operations: The backend interacts with MongoDB to perform CRUD (Create, Read, Update, Delete) operations.
  4. Response to User: The backend sends a response back to the frontend for rendering.
- 
- A decorative geometric pattern at the bottom of the slide, consisting of several overlapping triangles in shades of gray, creating a modern, abstract design.

# Prerequisites for Setting Up a Vite + React, Node.js, Express, and MongoDB Stack

Before diving into setting up your development environment for a full-stack application using Vite + React for the frontend and Node.js, Express, and MongoDB for the backend, make sure you have the following tools and knowledge:

## 1. Basic Knowledge & Skills:

**JavaScript/ES6+ Syntax:** Familiarity with modern JavaScript features like `async/await`, destructuring, arrow functions, template literals, etc.

**Node.js and Express:** Basic understanding of Node.js for running server-side JavaScript, and Express.js for building backend APIs.

**React:** Basic knowledge of React concepts such as components, JSX, state, and props.

**MongoDB:** Understanding of MongoDB as a NoSQL database and the ability to perform CRUD operations using MongoDB's shell or through libraries like Mongoose.

## 2. Required Tools & Software:

- Node.js** (LTS Version): Node.js is required to run both the backend (Express) and the frontend build tools (Vite).

Download from: [Node.js Official Website](#)

- npm** or **Yarn**: A package manager for JavaScript.
  - npm comes bundled with Node.js, or you can use yarn as an alternative.
- Check your version:
  - npm -v or yarn -v
- MongoDB** (Locally or Cloud):
  - For local development, you can install MongoDB or use MongoDB Atlas for a cloud-based database.
  - MongoDB Community Edition**: [MongoDB Download Center](#)
  - MongoDB Atlas** (Cloud): [MongoDB Atlas](#)
  - Code Editor**: A text editor like [Visual Studio Code](#) is recommended for development, though any editor will work.
  - Postman** (Optional): For testing your Express APIs, Postman or another API testing tool can be helpful.
  - Download from: [Postman](#)



### 3. Setting Up Your Development Environment:

#### Frontend Setup:

##### 1.Vite + React:

- Vite is a fast, modern build tool that provides excellent support for React applications.
- Create a Vite + React project:

```
bash
npm create vite@latest my-app --template react cd my-app npm install npm run dev
```

This will set up a basic Vite + React development server that you can access in your browser at <http://localhost:3000>.

**2.React Router (optional):** If you plan to handle routing within your React app, install React Router:

```
bash
npm install react-router-dom
```

#### Backend Setup:

##### 1.Node.js + Express:

- Create a new Node.js project for your backend API:

```
bash

mkdir my-backend cd my-backend npm init -y npm install express mongoose dotenv cors
```

- Set up a simple Express server in index.js:

```
const express = require('express'); const cors = require('cors'); const mongoose = require('mongoose');
require('dotenv').config(); const app = express(); app.use(cors());
app.use(express.json()); // MongoDB connection mongoose.connect(process.env.MONGO_URI, {
useNewUrlParser: true, useUnifiedTopology: true }) .then(() => console.log('Connected to MongoDB'))
.catch(err => console.log('Error connecting to MongoDB:', err)); app.get('/', (req, res) => { res.send('Hello from
Express!'); }); const PORT = process.env.PORT || 5000; app.listen(PORT, () => { console.log(`Server is running
on port ${PORT}`); });
```

Create a .env file in the root of your project for environment variables (e.g., MongoDB URI):

```
makefile
MONGO_URI=your_mongodb_connection_string
```

## 2.MongoDB with Mongoose:

- Mongoose is an ODM (Object Data Modeling) library for MongoDB and Node.js.
- Set up a basic schema/model:

```
js
const mongoose = require('mongoose'); const userSchema = new mongoose.Schema({ name: { type: String,
required: true }, email: { type: String, required: true, unique: true }, }); const User = mongoose.model('User',
userSchema); module.exports = User;
```

## 4. Connecting Frontend to Backend:

Once both the frontend (React) and backend (Express) are set up, you can connect them using RESTful APIs. For example, use fetch or Axios in React to make requests to your Express backend.

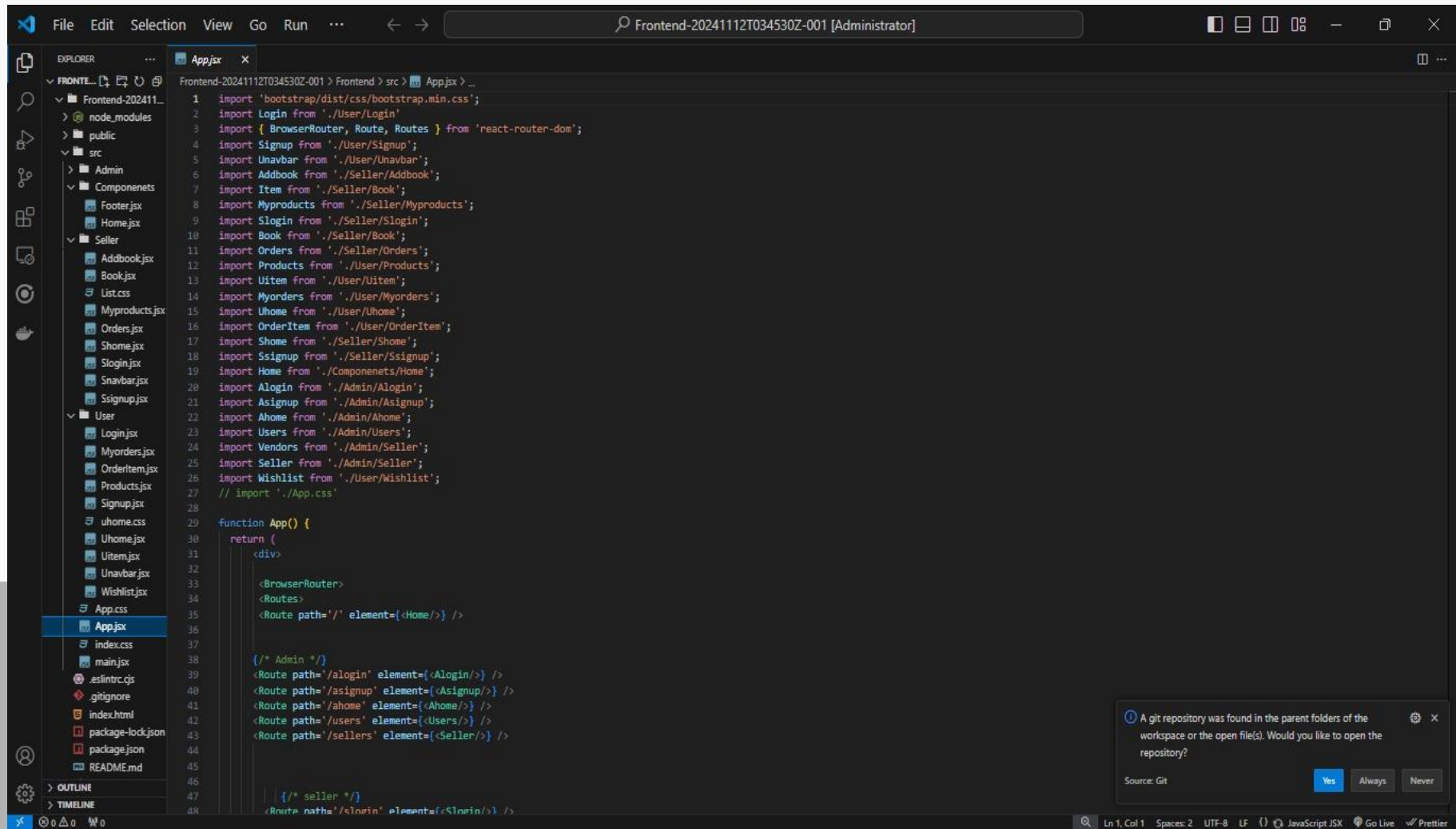
### Example of fetching data from Express backend:

js

```
In a React component: import { useState, useEffect } from 'react'; function App()  
  { const [message, setMessage] = useState("");  
    useEffect(() => { fetch('http://localhost:5000/') .then(response => response.text()) .  
      then(data => setMessage(data)); }, []);  
    return ( <div> <h1>{message}</h1> </div> ); }  
export default App;
```



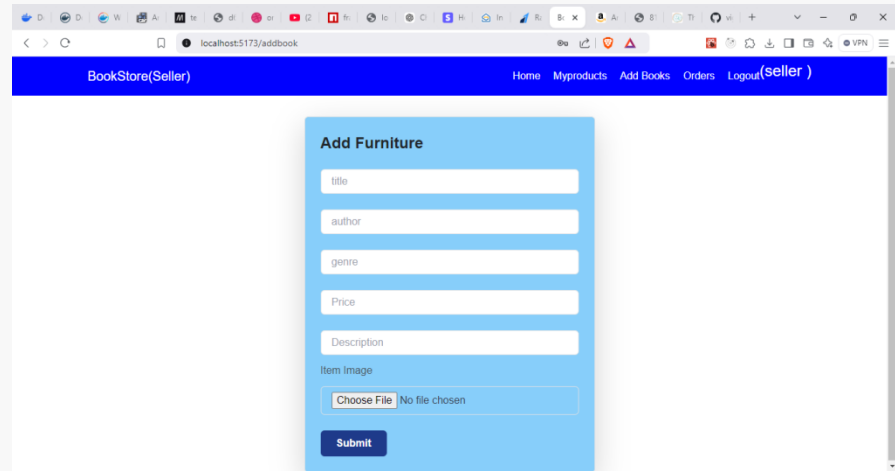
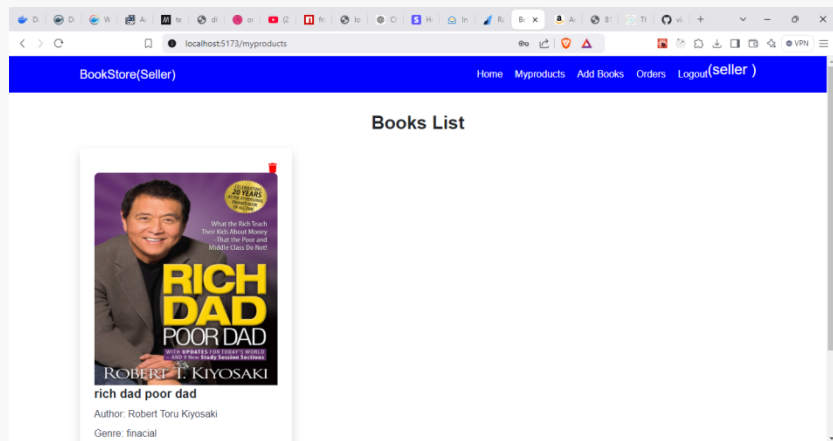
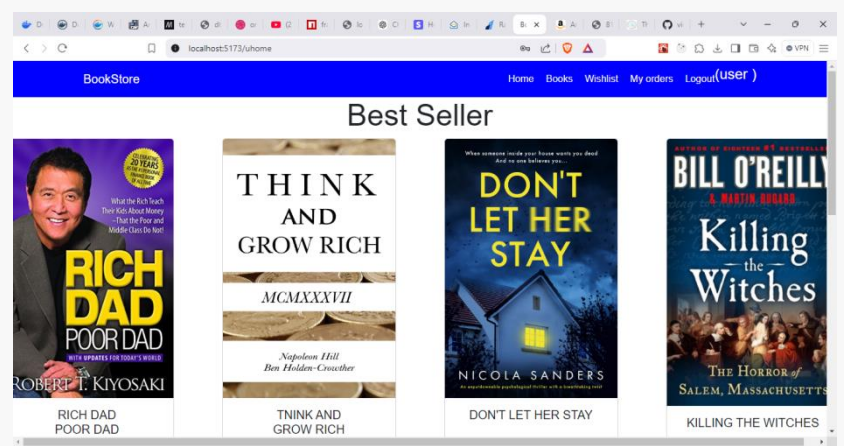
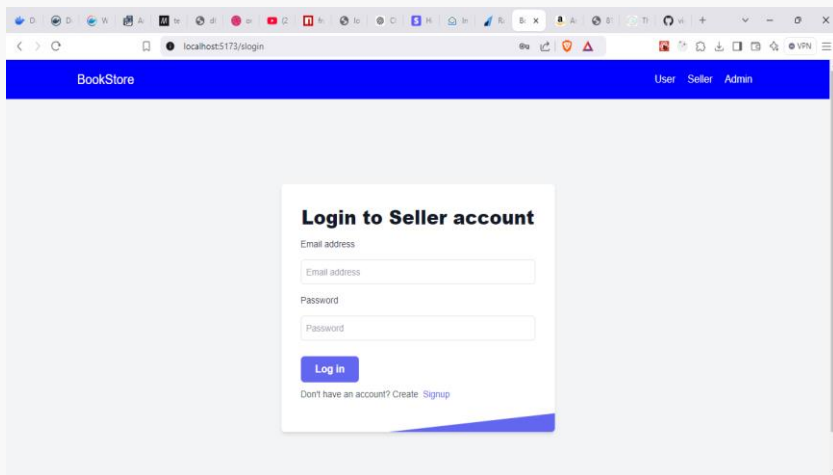
# Project structure:



## **Project Implementation and Execution**


The backend is built with Node.js and Express.js. Each route handles specific API requests such as retrieving books, adding books to the cart, and placing orders. MongoDB is used to store and query data about books, users, and orders.

The frontend is a simple HTML/CSS/JavaScript interface that interacts with the backend API to display books and handle the user flow (e.g., adding books to cart, user login).




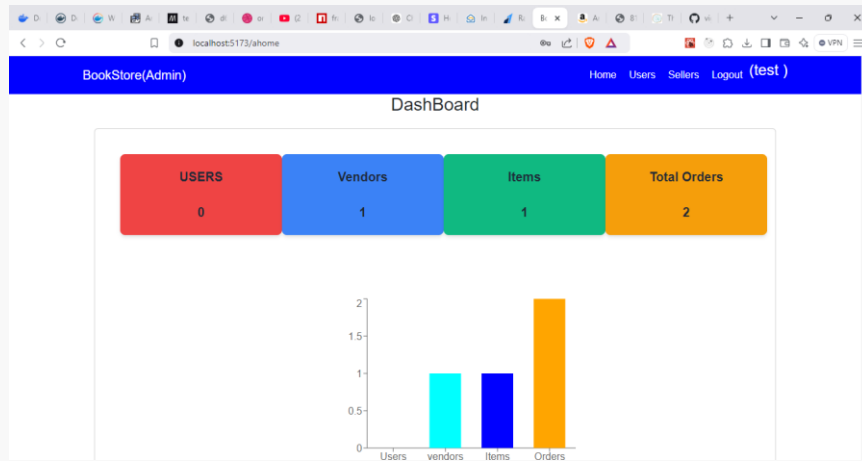
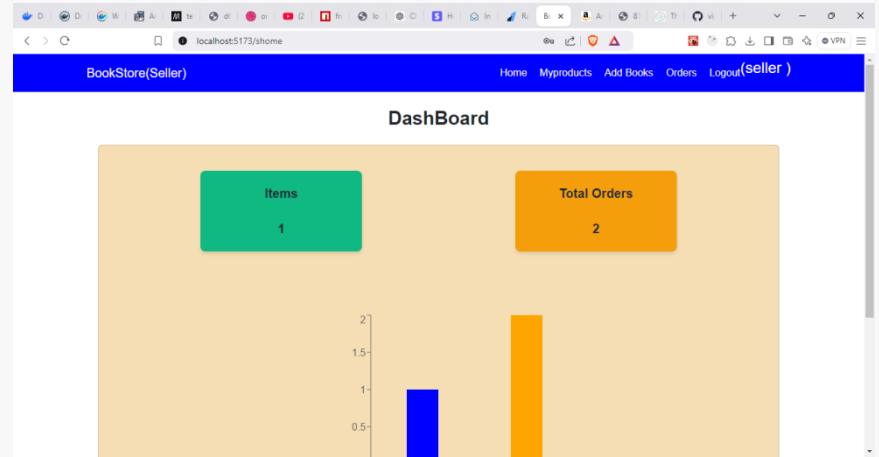
BookStore(Seller) Home Myproducts Add Books Orders Logout(seller )

### Orders

	ProductName:	Orderid:	Customer Name	Address:	BookingDate	Delivery By	Warranty	Price	Status
		6732d32fc1	user	chennai, chennai,(600002), tamilnadu.	12/11/2024	11/19/2024	1 year	798	ontheway

	ProductName:	Orderid:	Customer Name	Address:	BookingDate	Delivery By	Warranty	Price	Status
		6732d367c1	user	chennai, chennai,(600002), tamilnadu.	12/11/2024	11/19/2024	1 year	798	ontheway



## **Conclusion:**

The Book Store application demonstrates how a simple ecommerce platform can be built using MongoDB for storing and retrieving data. This architecture allows for flexibility and scalability as the system grows. By using MongoDB's documentbased structure, we can efficiently manage book data, user profiles, and orders. This project highlights the power of NoSQL databases for realworld applications that require high performance and scalability.



## **Demo Links :**

**Github Link:** [https://github.com/vigneshxcode1/nm\\_bookstore.git](https://github.com/vigneshxcode1/nm_bookstore.git)

# THANK YOU

