**BOOK STORE – MERN PROJECT**

**Abstract: Second-Hand Bookstore Web Application (MERN Stack)**

**This project is a second-hand bookstore web application built using the MERN stack (MongoDB, Express.js, React.js, Node.js). Unlike traditional e-commerce or publisher-based platforms, this system is tailored for individual users to buy and sell pre-owned books conveniently.**

**The platform is divided into User and Admin modules, each with distinct functionalities:**

**User Module Features:**

* **User Authentication:**
  + **Registration, Login, and Google OAuth Integration.**
* **Explore and Search Books:**
  + **Search functionality with filters.**
  + **"Latest Collection" section to view newly listed books.**
  + **Testimonials section to showcase user feedback.**
  + **“Explore More” feature for book recommendations.**
* **Book Management:**
  + **View books, buy books securely using Stripe payment gateway.**
  + **Sell books by uploading details and requesting admin approval.**
* **Profile Management:**
  + **Update personal profile.**
* **Job Opportunities:**
  + **View current job openings under "Careers".**
  + **Apply for jobs directly through the platform.**
* **Session Management:**
  + **Logout option to securely end sessions.**

**Admin Module Features:**

* **Admin Authentication:**
  + **Secure login system for administrators.**
* **Dashboard Management:**
  + **Overview of user and book statistics.**
* **Book Control:**
  + **View all listed books.**
  + **Approve/reject book selling requests from users.**
* **Job Management:**
  + **Add, update, and delete job listings.**
  + **View applications submitted by users.**
* **Profile Management:**
  + **Admin can update their own profile.**

**This application provides a user-friendly, secure, and dynamic platform for buying and selling second-hand books, making reading more affordable and sustainable. Through intuitive design and powerful features, it bridges the gap between book enthusiasts and reusable resources.**

**PROJECT CREATION STEPS**

**1 Create a Folder Book store**

**2 inside that folder create application (Create Vite + React Project)**

**npm create vite@latest bookstore-frontend -- --template react**

**cd bookstore-frontend**

**npm install**

**code .**

**3 Run the Development Server**

**npm run dev**

**4 Enable Auto Open in Browser**

1. **Open package.json**
2. **Find the scripts section**
3. **Edit the dev script like this:**

**"scripts": {**

**"dev": "vite --open",**

**"build": "vite build",**

**"preview": "vite preview"**

**}**

**5 Remove Unwanted codes**

**Delete Unwanted Files: Inside the src folder**

**6 Install 3rd-Party Libraries**

* 1. **Install Tailwind CSS v4** 
     1. **Link :** [**Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.**](https://tailwindcss.com/)
     2. **Install : npm install tailwindcss @tailwindcss/vite**
     3. **Configure the Vite plugin : Add the @tailwindcss/vite plugin to your Vite configuration.**

**vite.config.ts**

**import { defineConfig } from 'vite'**

**import tailwindcss from '@tailwindcss/vite'**

**export default defineConfig({**

**plugins: [**

**tailwindcss(),**

**],**

**})**

* + 1. **Import Tailwind CSS : Add an @import to your CSS file that imports Tailwind CSS.**

**Index.CSS**

**@import "tailwindcss";**

* + 1. **Start using Tailwind in your HTML – Inside index.css**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<link href="/src/styles.css" rel="stylesheet">**

**</head>**

* + 1. **Check Tailwind is working – App.jsx**

**<h1 class="text-3xl font-bold underline"> Hello world! </h1>**

* + 1. **Install & Use Flowbite with Tailwind CSS in Vite + React**

**Flowbite is a UI component library built on Tailwind CSS, offering pre-built components like modals, navbars, buttons, etc.**

**Link :** [**Flowbite - Build websites even faster with components on top of Tailwind CSS**](https://flowbite.com/)

**Install : npm install flowbite flowbite-react**

**Import Flowbite Styles**

**In src/index.css, add:**

**@import 'flowbite';**

* 1. **Install React Icons**

**Link :** [**React Icons**](https://react-icons.github.io/react-icons/)

**Install : npm install react-icons --save**

* 1. **Install React bootstrap**

**Link :** [**React Bootstrap | React Bootstrap**](https://react-bootstrap.github.io/)

**Install : npm install react-bootstrap bootstrap**

**Import Bootstrap CSS :**

**Go to main.jsx (or index.js for CRA apps), and import Bootstrap's CSS:**

**import 'bootstrap/dist/css/bootstrap.min.css';**

* 1. **Link Google fonts**

**Link :** [**Browse Fonts - Google Fonts**](https://fonts.google.com/)

* 1. **Toaster**

**Link :** [**https://www.npmjs.com/package/react-toastify**](https://www.npmjs.com/package/react-toastify)

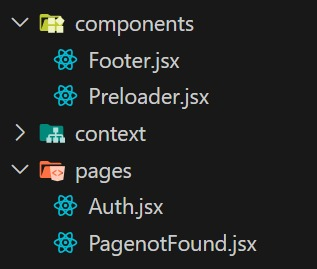
**JSON Authentication**

* + 1. **JWT**

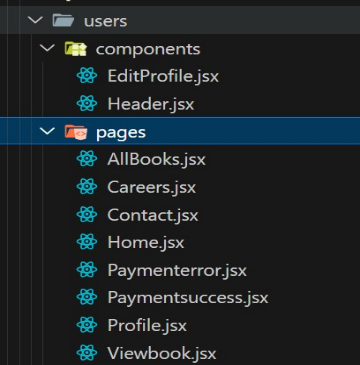
**Link :** [**https://www.npmjs.com/package/jsonwebtoken**](https://www.npmjs.com/package/jsonwebtoken)

**7 Component Creation**

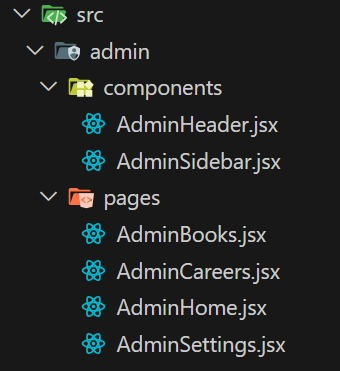
**Common Components and Pages**

****

**Users**

****

**Admin**

****

**8 React Router DOM Setup**

* + 1. **Install React Router DOM**

**npm install react-router-dom**

* + 1. **Set Up Basic Routing**

**import React from 'react'**

**import ReactDOM from 'react-dom/client'**

**import App from './App'**

**import { BrowserRouter } from 'react-router-dom'**

**import './index.css'**

**ReactDOM.createRoot(document.getElementById('root')).render(**

**<React.StrictMode>**

**<BrowserRouter>**

**<App />**

**</BrowserRouter>**

**</React.StrictMode>,**

**)**

**3. Use routes and route**

**import './App.css'**

**import { Route, Routes } from 'react-router-dom'**

**import Home from './users/pages/Home'**

**import Auth from './pages/Auth'**

**import PagenotFound from './pages/PagenotFound'**

**import Preloader from './components/Preloader'**

**import { useEffect, useState } from 'react'**

**import AllBooks from './users/pages/AllBooks'**

**import Careers from './users/pages/Careers'**

**import Contact from './users/pages/Contact'**

**import Profile from './users/pages/Profile'**

**import AdminHome from './admin/pages/AdminHome'**

**import AdminBooks from './admin/pages/AdminBooks'**

**import AdminCareers from './admin/pages/AdminCareers'**

**import AdminSettings from './admin/pages/AdminSettings'**

**import Viewbook from './users/pages/Viewbook'**

**import Paymentsuccess from './users/pages/Paymentsuccess'**

**import Paymenterror from './users/pages/Paymenterror'**

**function App() {**

**const [isloading, setIsloading] = useState(false)**

**useEffect(() => {**

**setTimeout(() => {**

**setIsloading(true)**

**}, 7200)**

**}, [])**

**return (**

**<>**

**<Routes>**

**<Route path='/' element={isloading ? <Home /> : <Preloader />} />**

**<Route path='/login' element={<Auth />} />**

**<Route path='/register' element={<Auth register />} />**

**<Route path='/all-Books' element={<AllBooks />} />**

**<Route path='/view-books/:id' element={<Viewbook />} />**

**<Route path='/careers' element={<Careers />} />**

**<Route path='/contact' element={<Contact />} />**

**<Route path='/profile' element={<Profile />} />**

**<Route path='/admin-home' element={isloading ? <AdminHome /> : <Preloader />} />**

**<Route path='/admin-books' element={<AdminBooks />} />**

**<Route path='/admin-careers' element={<AdminCareers />} />**

**<Route path='/admin-settings' element={<AdminSettings />} />**

**<Route path='/payment-success' element={<Paymentsuccess />} />**

**<Route path='/payment-error' element={<Paymenterror />} />**

**<Route path='\*' element={<PagenotFound />} />**

**</Routes>**

**</>**

**)**

**}**

**export default App**

**BACKEND CREATION**

**1 Create a folder named as Server (bookstore-backend)**

**2 Create package.json file – npm init**

**3 Install 3rd party libraries**

* **Express :used for creating server :** [Express - Node.js web application framework](https://expressjs.com/)
* **Mongoose : For connecting mongodb with nodeJs :** [Mongoose ODM v8.16.3](https://mongoosejs.com/)
* **Cors : (CROSS ORIGIN RESOURCE SHARING) :For Connecting frontend port :** [Handling CORS with Node.js](https://stackabuse.com/handling-cors-with-node-js/)
* **Dotenv : Handle enviounment variables :** [Node.js Environment Variables](https://www.w3schools.com/nodejs/nodejs_environment.asp)

[Documentation | Dotenv](https://www.dotenv.org/docs/)

**npm i express mongoose cors dotenv**

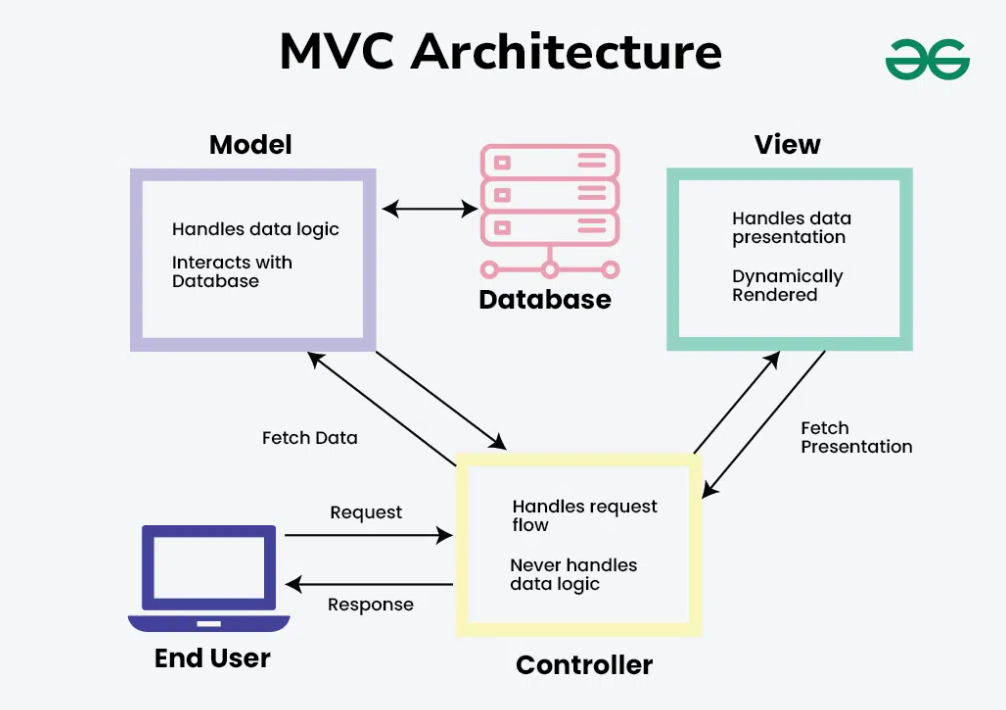
**4 Create .env and .gitignore files**

**5 Create main file – index.js**

**6 Middlewares :** [Using Express middleware](https://expressjs.com/en/guide/using-middleware.html)

[Node.js Middleware Concept](https://www.w3schools.com/nodejs/nodejs_middleware.asp)

7 **MVC(Model-View-Controller) : Architecture is a fundamental design pattern in software development, separating an application into Model, View, and Controller components. This article explores its role in building robust, maintainable systems, emphasizing its benefits and implementation strategies.**



Model: It is worth stating that the Model stands as the data layer for the application. It is directly involved in managing the data as well as the control of the application’s logic and rules.

View: The View is in the presentation tier. It plays a role of presenting the information given by the Model to the user and transferring the user commands to the Controller. The View is used to display the data to the user in a readable and manageable way using the interface created by the Controller.

Controller: The Controller CE works in the middle between the Model and the View. It takes the input from the View, sometimes modifies it with the help of the Model, and sends it back to the View. the results back to the View.

1. **Create MongoDB Database**

* Set up your MongoDB database (locally or on MongoDB Atlas).
* Define the necessary collections based on your application needs.

2. **Create Model and Schema**

* In the models folder, define the data structure using Mongoose schemas.
* These models represent the structure of data stored in your database.

3. **Create Controller**

* In the controllers folder, write the logic to handle requests (e.g., fetching, saving, updating data).
* Controllers interact with models and send appropriate responses.

4. **Create Router**

* In the routes folder, define routes using Express Router.
* Connect each route to the corresponding controller function.
* This helps organize routes related to different features (e.g., users, posts, products).

5. **Connect Everything in index.js**

 Load environment variables using dotenv (e.g., DB connection URL, port).

 Connect to the MongoDB database using the connection string from .env.

 Apply middlewares like express.json() for JSON parsing.

 Import and use route files with appropriate base paths (e.g., /api/users).