Manga Mania

Implementation Document

Jodan Elysee

Abstract

This paper delves into the development of an online manga-selling website using the MERN (MongoDB, Express.js, React, Node.js) Stack, a popular framework for building full-stack web applications. The paper will also discuss the planning, setup, and implementation of the application, including user management, MongoDB collections structure, React component design, page layout, routing, and styling. It explores additional tools such as Vite.js for enhanced development features, Tailwind CSS for streamlined styling, Flowbite React for UI component integration, and Swiper.js for interactive slideshow components.

I: Introduction

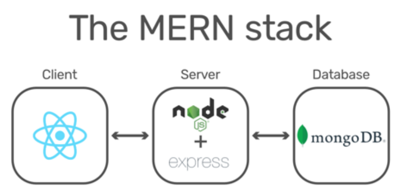

Figure 1: Diagram of MERN Stack

II: MERN Stack

In the MERN Stack, MongoDB serves as the database layer, Express.js handles server-side logic and routing, React manages the client-side user interface, and Node.js provides the runtime environment for running JavaScript on the server side. Together, these components enable developers to build full-stack web applications using JavaScript across the entire stack, offering a seamless and consistent development experience.

A: MongoDB

MongoDB is a NoSQL database that stores data in a JSON-like format. It is known for its scalability, flexibility, and ability to handle substantial amounts of data. MongoDB uses collections and documents instead of tables and rows, making it a good fit for applications with dynamic and evolving data structures.

B: Express.js

Express.js is a minimalist web framework for Node.js. It simplifies the process of building web applications and APIs by providing a set of robust features for handling HTTP requests, routing, middleware, and more. Express.js is known for its flexibility and ease of use, making it a popular choice for backend development in the MERN Stack.

C: React

React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components and manage the state of an application efficiently. React uses a virtual DOM (Document Object Model) to optimize rendering performance, making it suitable for building interactive and dynamic web interfaces.

D: Node.js

Node.js is a runtime environment that allows developers to run JavaScript code outside the web browser, typically on the server side. It uses an event-driven, non-blocking I/O model, which makes it lightweight and efficient for handling concurrent requests. Node.js is commonly used in the MERN Stack for server-side logic, data processing, and handling database operations.

III: Additional Tools

A: Vite JS

Vite

Vite.js is a modern build tool and development server that enhances the development experience for React applications. It leverages ES modules for fast and efficient code bundling, provides a development server with hot module replacement (HMR) for instant code updates, and optimizes production builds for smaller bundle sizes and faster loading times. Vite.js also supports modern CSS features.

B: Tailwind CSS

tailwindcss

Tailwind CSS is a utility-first CSS framework that streamlines the styling process for React developers by providing a set of pre-designed utility classes. Its utility-first approach eliminates the need for writing custom CSS styles, allowing developers to apply styles directly in their HTML or JSX code. Tailwind CSS offers responsive design capabilities, customization options, and seamless integration with React projects.

C: Flowbite React

Flowbite React

Flowbite React is a UI component library that integrates with Tailwind CSS and React, offering a collection of pre-designed UI components and styles for building web applications. It promotes component-based development, aligning with React's architecture, and allows for customization and extensibility through Tailwind CSS classes. Flowbite React enhances developer productivity by providing ready-to-use components and seamless integration with Tailwind CSS

D: Swiper JS

Swiper

Swiper.js integrates seamlessly with React projects, allowing developers to build interactive and visually appealing slideshow components with smooth touch interactions and efficient performance.

IV: Application

The Web application that I am presenting is an online manga-selling website where users can buy and sell books. Manga is a Japanese-style comic book. Users will be able to upload books at any price they want others to purchase. This idea is a mix between eBay and Barnes & Noble.

A: Planning

When it came to planning for this project I had to decide between two frameworks. The decision was between using PHP with CodeIgniter or JavaScript with the MERN Stack. After researching both, I went with the MERN Stack. I picked the MERN stack due to the document style database, MongoDB, and to try out something new since I have not used the MERN Stack and all its elements before.

After selecting the framework, I had to plan out the relationships and the functionality of the application. I had to make an ERD (Entity-Relationship Diagram) to help me visualize how the data in MongoDB would look. This also helped me figure out what attributes each element would have.

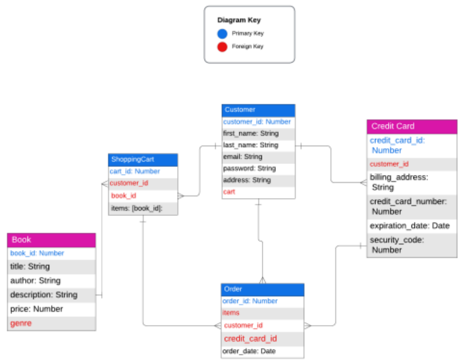

Figure 2: Entity Relationship Diagram

B: Setup

To set up my application, I had to implement and install all the components of the MERN Stack along with all the additional components. Using the node package installer (npm) I installed node.js and express.js. I also installed nodemon which allows me to run the backend (node server) automatically and it will restart every time I edit my code. Next, I created an Atlas account so that I could use MongoDB. After signing up, I made a new cluster, and within that cluster, I made a new database. I allowed the database to be accessed from any IP and created a new user that has access to everything about the database. After setting up the database, I connected the database to my code using the connection string. After the backend setup, I started working on the front end (client side). I installed Next.js, which is a React framework, using npx (node package eXecute) to create a new React project. After the setup, I was ready to begin working on the backend.

C: Users

For my application, there will be users who all have the same ability to upload and purchase books. Each user will have their own set of credentials: first name, last name, email, password, and address. Users will also be able to add, edit, and delete credit/debit cards. Users will also be able to register, log in, add manga to their cart, and view their order history. Users will also be able to change and update their credentials. Using a useEffect, if you are not logged in, you will not be able to access any of these features.

D: MongoDB

To manage all my data, I used MongoDB as my database. I will hold all my data in 5 collections. I made a collection for users, cards, carts, orders, and books. Each book will have a title, author, price, image URL, genre, description, and a reference to a user. Each card will have a card number, expiration date, security code, owner name, billing address, and a reference to a user. Each cart will have a reference to a user and an array that carries book IDs. Each order will have a reference to a user and an array that carries book IDs.

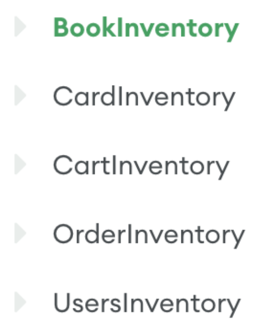

Figure 3: MongoDB Collections

E: React Components

For my react components, I created a file for each component I will be creating. I made a file for the about page, the banner on the homepage, the cards to hold the books on the homepage, the cart page, the login page, the registration page, the footer at the bottom of each page, the customer reviews at the bottom of the homepage, the navbar that will contain links that redirect to every page, the search bar on the homepage, the orders page, the shop page where you can view all of the books, and the dashboard along with all of its components.

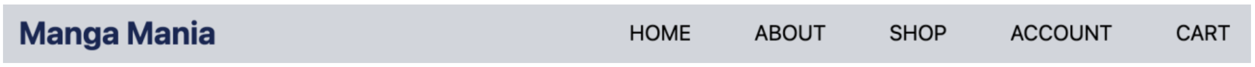

Figure 4: Navbar Component

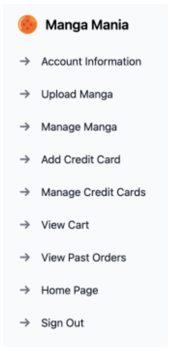

Figure 5: Dashboard Sidebar Component

F: Pages

The application will consist of all pages I mentioned in the previous section. The first page that the users will encounter is the login page. Users with credentials will be able to log in to the site and access all pages, if users do not have login credentials, there will be a link for users to register to gain credentials to be able to log in. After logging in, users will be redirected to the homepage, and there will be a navbar that will have links that redirect users to the specific page that they want to go to. Using the navbar users will be able to navigate to the homepage, the shop page, the account page, and the cart. On the account page, the sidebar has links to navigate to the account information page, the upload manga page, the manage manga page, the add credit cart page, the manage credit cards page, the cart page, the past orders page, the home page, and the sign in page. If users try to go to a page that does not exist, a 404 error will appear telling the user that the page was not found because it does not exist.

G: Routing

To manage the route to go to each page, I am importing BrowserRouter as createBrowserRouter, from react-dom-router into my router.jsx file. I also imported the functions from each of the components. In my router function, I created routes for every page and for some of the routes, I am using a loader to fetch data from the backend.

I: Styling

To keep the website from looking bland, I decided to use additional tools along with React to achieve this. I used Tailwind CSS, which is a CSS framework, along with Flowbite React and Swiper JS. I described these tools in the Additional Tools section. I also imported 4 image assets: cherryblossom.png to appear on the promo banner component, dragonball.png to show in the sidebar component, FavBooks.png to show in the favorite books component, and homePage.jpeg to show on the about page.

J: Running the application

To run the application, inside the terminal, you will need to redirect to the backend folder and run the command “npm start.” This will connect to MongoDB and get all the data from the collections. The next step is to use the r terminal redirect to the mern-client folder and use the command “npm run dev.” This command is used to execute specific scripts defined in the package.json file, particularly those scripts related to the development environment setup. It also makes it easier to inspect any errors in the developer tools in the console in your browser.

V: Conclusion

After building this application, I have received a better understanding of MERN Stack and each of its components. Using this software framework has made it easier to complete this application. Also, using a styling framework along with React has improved my knowledge of CSS styling. I plan to use the MERN Stack to add more to this project soon and I will use the MERN Stack on more applications in the future.

References

I. Tailwind CSS. Tailwind CSS, 2024, https://tailwindcss.com/.

II. Swiper. Swiper, 2024, https://swiperjs.com/.
III. Vite. Vite, 2024, https://vitejs.dev/guide/.
IV. "What is the difference between npm run dev and npm run production? - Laracasts." Laracasts, 2024, https://laracasts.com/discuss/channels/javascript/what-is-the-difference-between-npm-run-dev-and-npm-run-production.
V. "What are npm run dev and npm run prod?" Stack Overflow, 2024, https://stackoverflow.com/questions/64397005/what-are-npm-run-dev-and-npm-run-prod.