Proshop E-commerce Application

Description

Proshop is a fully fledged ecommerce application that showcases products setup by admins for clients to buy. It has the functionality of full featured shopping cart, Product reviews and ratings, Top products carousel, Product pagination, Product search feature, User profile with orders, Admin product management, Admin user management, Admin Order details page, Mark orders as delivered option, Checkout process (shipping, payment method), PayPal integration.

Advantages of Technologies Used

The application was built with ReactJs as the frontend, state management using Redux, backend using NodeJs lightweight Express framework (followed RESTful pattern), authentication using JSON web tokens(jwt), database uses MongoDB Atlas cloud database.

(a) Frontend using ReactJs

According to Netflix,React has a one-way data flow and declarative approach to UI development makes it easier to reason about our app.

I wanted to build a single page application meaning the application should be able to update certain parts of itself without reloading the page. This ensures a good user experience. A great benefit of using reactjs is the ability to write code as components and reuse them in other parts of the application. React also has a great community which makes searching for solutions to problems encountered during development easy.

I used redux to manage the state of the application since it gives me the ability to have a single source of truth that is global to the whole application.

(b) Backend using express js

MDN web doc defines express js as a popular unopinionated web framework, written in JavaScript and hosted within the Node.js runtime environment.

It gives freedom in building an application backend how you want it. It provides the bare necessities for backend development. I used expressjs because I wanted to have more control on the setup and development process but get to add packages that will need to make the development easy via npm. An example is me using the JWT package to handle authentication when I needed it. With other frameworks they come already configured with authentication irrespective of the fact that you may not use it, this can make the app unnecessarily huge and may lead it to become slow in production.

(c) MongoDB for Database

Setting up mongodb is very easy.As compared with SQL databases, mongodb is very easy to work with and it also has the mongoose library which makes creating documents and querying them very easy. With the mongodb cloud, I was able to host my database online so easily and for free.

Code Structure

The code is put into two main folders, client and backend which are all in the root application. The client contains the react app and the backend contains the expressjs app which houses the database config, controllers, routes, middleware, utils and seeding data . The root contains the package json and lock, .env, and readme.md

```
Proshop
├── backend
│     ├── config
│     │      └── db.js
│     │
│     ├── controllers
│     │      ├── OrderController.js
│     │      └── productController.js
│     │      ├── userController.js
│     │      └── productController.js
│     ├── data
│     │      ├── product.js
│     │      └── users.js
│     ├── middleware
│     │      ├── authMiddleware.js
│     │      └── errorMiddleware.js
│     ├── models
│     │       ├── OrderModel.js
│     │       ├── ProductModel.js
│     │       └── UserModel.js
│     ├── routes
│     │        ├── OrderRoutes.js
│     │        ├── ProductRoutes.js
│     │        ├── uploadRoutes.js
│     │        └── UserRoutes.js
│     ├── utils
│     │        └── generateToken.js
│     ├── seeder.js
│     └── server.js
├── client
       ├── README.md
│      ├── node_modules
│      ├── package.json
│      ├── .gitignore
│      ├── public
│           ├── favicon.ico
│           ├── index.html
│           └──images
       └── src
```

```
│            ├── components/
│            ├── screens/
│            ├── store
│                  ├── Actions
│                  ├── reducers
│                  ├── store.js
│                  ├── types.js
│
            ├── App.css
│           ├── App.js
│           ├── App.test.js
│           ├── index.css
│           ├── index.js
│           ├── logo.svg
            └── serviceWorker.js
├── node_modules
├── .gitignore
├── .env
├── package.json
├── package-lock.json
└── README.md
```

The backend is separated into models, routes and controllers to ensure clean architecture and separation of responsibilities. This is to ensure that app is easily tested and codebase can be easily expanded without damaging other parts of the application.

The Client has a screen folder which contain the various pages in the application, the components folder which contains the reusable parts of the app and the store which houses the redux part of the application, which is the actions, reducers and redux store.

Things Not Good about my project

1. The project is written in js which does not provide type safety.
2. Tests are not written for the application hence if future code is added and it breaks a part of the application, it might go unnoticed.
3. MongoDB might not scale well when the users grow so much.(Ongres conducted a performance test and found postgresql 4 to 15 times faster than MongoDB)
4. The project does not have code formatting and linting setup thus collaborating with other developers on the project can become tough.

5. Project was not built with microservice architecture in mind so scaling will become difficult in the future.

Things to make the code better

1. Write code in typescript
2. Write tests for the application
3. Setup eslint and prettier
4. Design application using microservice instead of monolith

Ways App may have been implemented

1. Use wordpress theme to design the app and handle buying with woocommerce/
2. Build the UI with reactjs and then handle the backend using a headless CMS such as graphcms or strapi.

References

1. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
2. https://netflixtechblog.com/crafting-a-high-performance-tv-user-interface-using-react-3350e5a6ad3b?gi=e1bcfb5e5ae8#:~:text=We%20decided%20to%20use%20React,that%20targeted%20Gibbon%20pretty%20quickly.
3. https://www.ongres.com/blog/benchmarking-do-it-with-transparency/