Project Report

18CSC310J- Data-Centric Networking and System Design
Academic Year: 2022 -2023

Title: E-commerce Website Using MERN

By

DURGA CHANDANA SREE M(RA2011028010099)

ROSHINI DEVI JAMMULA (RA2011028010104)

REDDY JYOTHI SRI D (RA2011028010108)

Under the Guidance of

Dr. D Saveetha

Associate Professor, Department of Networking and Communication

SCHOOL OF COMPUTING

Department Of Networking & Communications

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur – 603203

INDEX

Objective:

To build an e-commerce application using React, which is inspired by Amazon. The latter part is to create a login and Registration of User pages using the firebase database enclosed with a Payment gateway.

Project Overview:

An E-commerce website is made which is an amazon clone using MERN. It has a Login and Registration section which is maintained by the database MongoDB and also has a payment gateway using Stripe API. Our clone will allow users to register, log in, add products to the shopping cart, and remove products from the shopping cart. Our clone will permit only an authenticated user to access the home page.

Abstract:

An E-commerce website is made which is an amazon clone using MERN. It has a Login and Registration section which is maintained by the database MongoDB and also having a payment gateway using Stripe API.

The first task, once we get the development environment ready, will be to set up the React Router. Once we have everything in place, we can start off with creating the website header, which will basically serve as the navigation bar, as in most modern websites. Next up is the home page building. In this project, we'll be keeping it simple by showing all our sample products on the homepage.

Then we'll be setting up the React Context API. This is a component structure provided by the framework, which enables us to share specific states across all levels of the application. In our project, we'll need to manage two states: basket (to manage the shopping cart) and user (for managing the details of the currently logged-in user).

For setting up the payment functionality, we'll be using APIs provided by Stripe. Handling our database and authentication needs to be supported and we'll be using it for the same. Basically, the database will be used to store the login information for the users, but the resource can be used for storing product information as well.

Once we have set up, we can work on the Login page of our application. Successful implementation of the above requirements will lead to the completion of the core implementation of our e-commerce solution. Next up, deploy!

In the end, we'll have a result similar to the one shown below.

Project Specifications:

Pre-requisite:

- Basic understanding of JavaScript ES6
- Basic understanding of HTML and CSS
- Have installed on your machine

Technologies:

- React
- Firebase
- Font awesome (icons)

Project setup:

Let's get started as we create a new project using the create-react-app so go to a directory where you will store this project and type the following in the terminal.

Create-react-app amazon-clone

"Bash"

The above command uses the create-react-app tool to generate a react boilerplate project for us so that we don't have to configure any tooling. CLI tool to generate a react boilerplate project for us so.

For the command above to work, the create-react-app CLI tool must be installed globally using the command below.

npm install -g create-react-app

Spin up the server with the following:

npm run start

We will delete some files that are not necessary for this project. These files include (App.css, App.test.js, logo.svg and registerServiceWorker.js)

Next, make the following change to the index.js file.

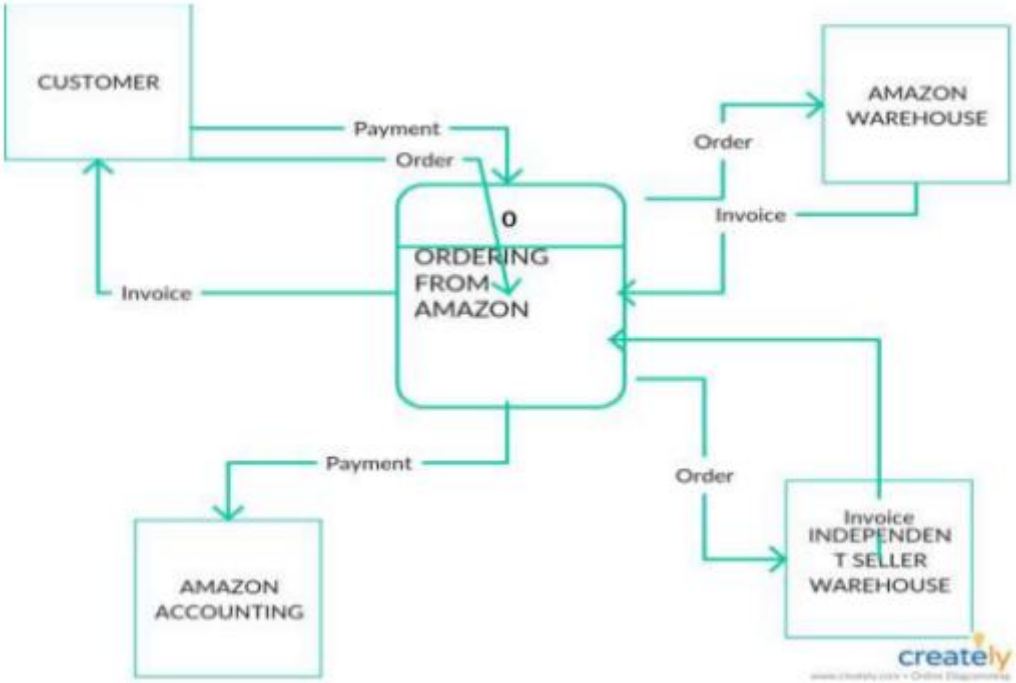
```
ReactDOM.render(
  <React.StrictMode>
    <StateProvider initialState={initialState} reducer={reducer}>
      <App />
    </StateProvider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

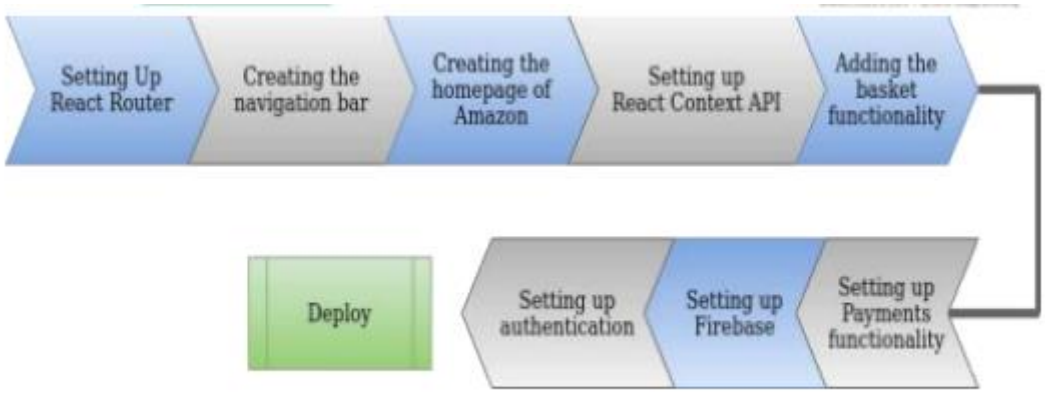
Components:

Our amazon clone will consist of the following components:

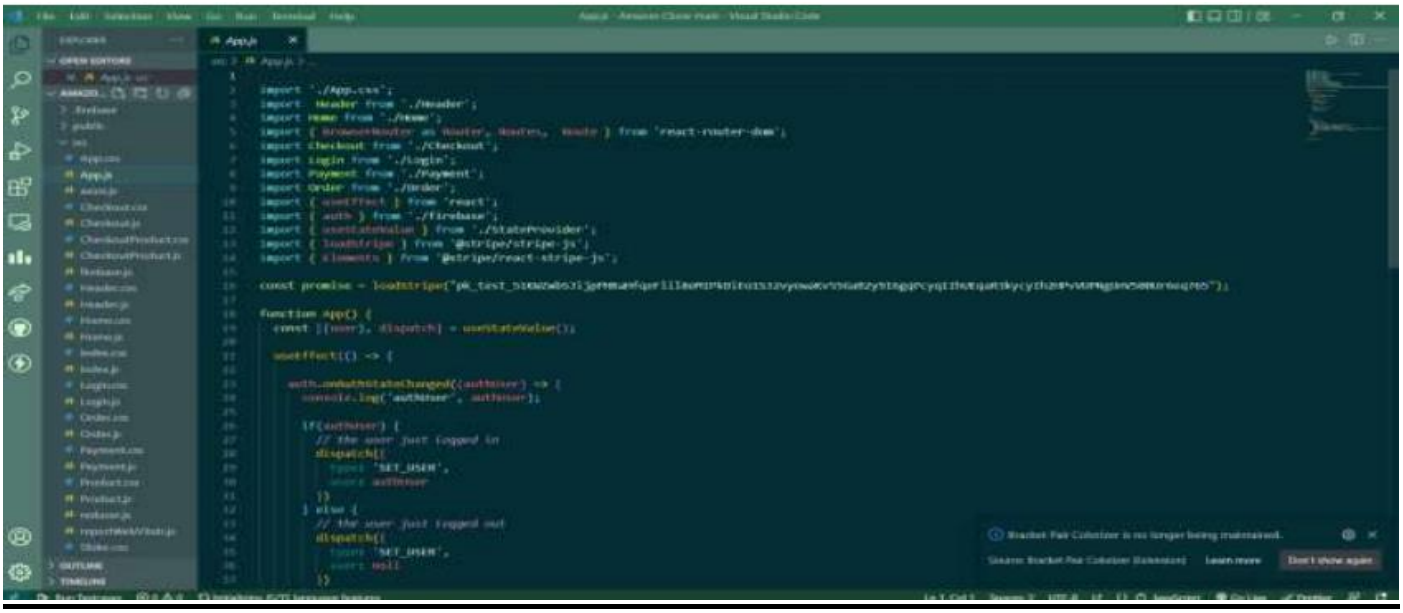
- Header
- Login
- Product
- Checkout
- Home
- Payment
- Order
- Subtotal

Design and UseCase Diagram:





Project Code:



App.js Component

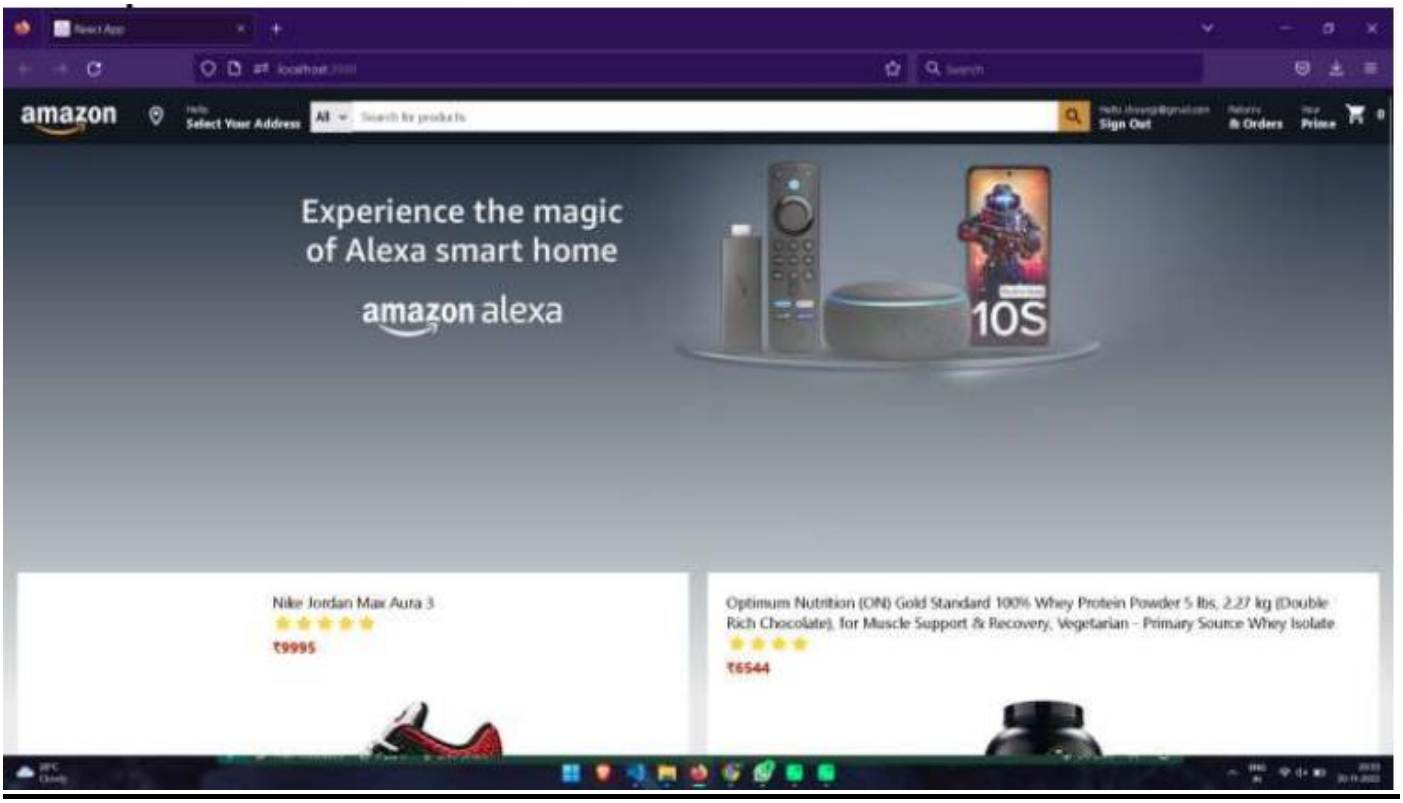
```
import './App.css';
import  Header from './Header';
import Home from './Home';
import { BrowserRouter as Router, Routes,  Route } from 'react-router-dom';
import Checkout from './Checkout';
import Login from './Login';
import Payment from './Payment';
import Order from './Order';              s
import { useEffect } from 'react';
import { auth } from './firebase';
import { useStateValue } from './StateProvider';
import { loadStripe } from '@stripe/stripe-js';
import { Elements } from '@stripe/react-stripe-js';

const promise =
loadStripe("pk_test_51KWZwbSJijpPHBaHfqeFlil8eMIPkDiEo1S32vyewaKv55GaB2y9I6gqPcyqIIhUEqa
R3kycyIh2HPvVUPNgUnV500Ur6eq765");

function App() {
  const [{user}, dispatch] = useStateValue();

  useEffect(() => {

    auth.onAuthStateChanged((authUser) => {
      console.log('authUser', authUser);

      if(authUser) {
```

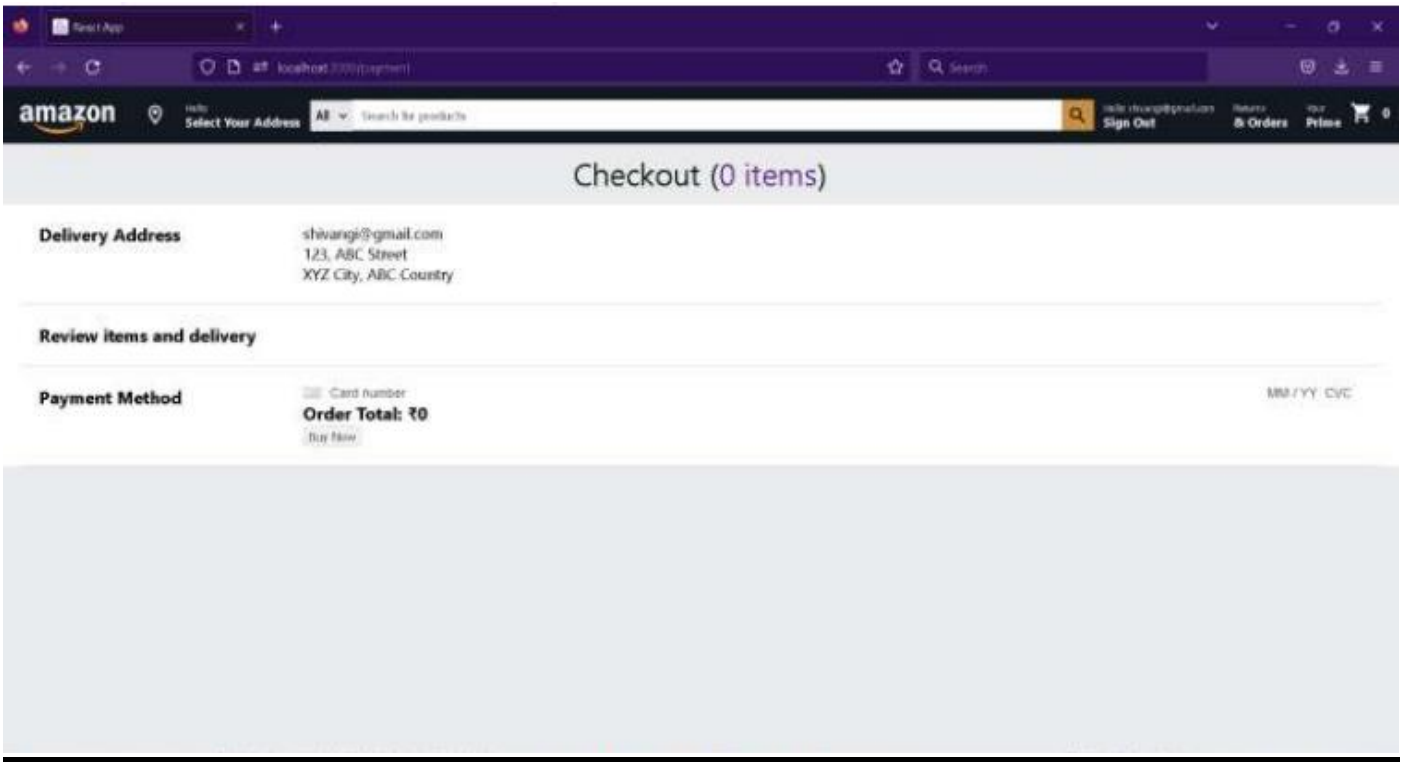
```
        // the user just logged in
        dispatch({
          type: 'SET_USER',
          user: authUser
        })
      } else {
        // the user just logged out
        dispatch({
          type: 'SET_USER',
          user: null
        })
      }
    })
  } , []);

  return (

    //BEM Convention
    <Router>
      <div className="App">
        <Routes>
          <Route path="/orders" element={[<Header />,<Order />]} />
          <Route path="/" element={[<Header />,<Home />]} />
          <Route path="/login" element={<Login />} />
          <Route path="/payment" element={[<Header />,<Elements
stripe={promise}><Payment /></Elements>]} />
          <Route path="/checkout" element={[<Header />, <Checkout/>]} />

        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

Firebase Component

```
import firebase from "firebase/compat/app";
import "firebase/compat/auth"
import "firebase/compat/firestore"


const firebaseConfig = {
  apiKey: "**********************************",
  authDomain: "dbms-b220b.firebaseapp.com",
  projectId: "dbms-b220b",
  storageBucket: "dbms-b220b.appspot.com",
  messagingSenderId: "865468113094",
  appId: "1:865468113094:web:31b6bd465b1962783fb7c7",
  measurementId: "G-CXH33FRZ7Z"
};

const db = firebase.initializeApp(firebaseConfig).firestore();
const auth = firebase.auth();

export { db, auth };
```

Demo and Screenshots:




amazon
Select Your Address
Sign Out
Checkout (0 items)
Delivery Address
shivangi@gmail.com
123, ABC Street
XYZ City, ABC Country
Review items and delivery
Payment Method
Card number
MM / YY CVC
Order Total: ₹0
Buy Now

Conclusions:

We have succeeded in building our Amazon clone using the Context API for state management and firebase for user authentication.

In the process, we have learned:

1. What the Context API is and the problem it solves

2. When to use the Context API

3. Creating and consuming it functional components.

4. What React Hook is

5. How to setup firebase authentication. I

n the end, most e-commerce operators will find that databases are what provide their customers the accessibility they demand. When product catalogs, online services, automated email responses, user feedback systems are managed via a database everyone is happier. MongoDB is a popular NoSQL database solution that suits modern development requirements.

References:

1. https://www.crio.do/projects/react-amazon-clone/
2. https://github.com/topics/amazon-clone
3. https://www.linkedin.com/pulse/amazon-clone-using-reactjs-ultimate-guide-fullstack-nirajan-mahara/?trk=portfolio_article-card_title
4. https://www.amazon.com/
5. https://www.studocu.com/in/document/government-polytechnic-nagpur/computer-engineering/me-project-amazon-clone-react/25536359