# AN E-COMMERCE SHOPPING WEBSITE FOR MOBILE PRODUCTS

Developed By

Name: R. Yazhinni

Registration Number: R201174601398

NIIT Anna Nagar

Title: An E-commerce shopping website for mobile products

Year: 2020

Language: English

Pages: 31

Name of Co-ordinator: Ms. Indumathi

Date of Submission: 31-08-2020

# ABSTRACT

The project aimed to develop an online shop using an open-source JavaScript library (ReactJs) for mobile products where customers will be able to buy products online. The customers will be able to view the list of products, their price, their brands, and they can add or remove it from the cart.

**TABLE OF CONTENTS**

## LIST OF FIGURES

# SYNOPSIS

# AN E-COMMERCE SHOPPING WEBSITE FOR MOBILE PRODUCTS

## 1. INTRODUCTION

The purpose of this project is that any person can sell their product online and earn cash. E-commerce (electronic commerce or EC) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the internet. These business transactions occur either as business-to-business, business-to-consumer, consumer-to-consumer, or consumer-to-business.

E-commerce shops have become part of our daily lives. Technological advancement has made it possible for people to sit in the convenience of their homes and still shop online without going to a physical shop.

The customer can search for products selection, update the cart, remove products from the cart and check out from the shop. The User is only able to browse the online shop and add a product to the cart. The user is limited to the use of the shop.

**Advantages of E-commerce:**

- Faster buying/selling procedure, as well as easy to find products.
- Buying/selling 24/7.
- Low operational costs and better quality of services.
- Customers can easily select products from different providers without moving around physically.

# 2. TOOLS AND TECHNOLOGIES

Framework used: ReactJs

IDE: Visual Studio Code

## 2.1. ReactJs:

- React is a JavaScript library for building user interfaces.
- React is used to build single page applications.
- React allows us to create reusable UI components.

## 2.2. Visual Studio Code:

- Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS.
- Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

# 3. INSTALLATION

- In order to learn and test React, we should set up a React Environment on our computer.
- To set up an environment for successful React development we need NodeJS and npm to be installed.

## 3.1. Setting up a React Environment:

1. Install create-react-app by running this command in your terminal:

```
C:\Users\Your Name> npm install -g create-react-app
```

2. To create a React application named store-recording run this command:

```
C:\Users\Your Name> npx create-react-app store-recording
```

3. Run this command to move to the store-recording directory:

```
C:\Users\Your Name> cd store-recording
```

4. Run this command to execute the React application store-recording:

```
C:\Users\Your Name\store-recording> npm start.
```

The customer can make changes to their data displayed on the customer page.

# 4. APPLICATION AND DESCRIPTION

## 4.1. Customer Detailed Attribute

- **View**

The customer can see all the products in the catalogue and able to look at the products and some features on the homepage.

- **Edit**

The customer can make changes to their data displayed on the customer page.

- **Update Cart**

This refers to putting or removing products from the shopping cart.

# 5. MVC UNIT OF SHOPPING CART

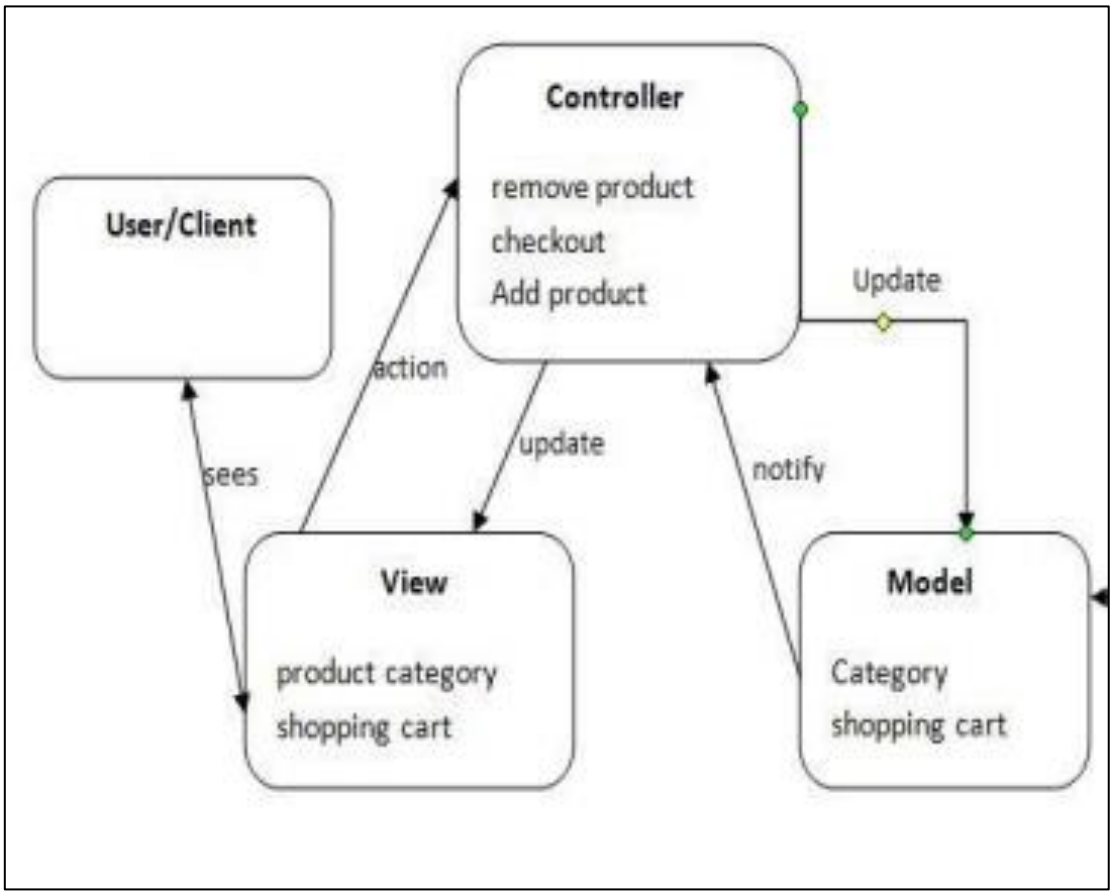


**Figure 1: MVC diagram of the online shop.**

The three parts of the MVC software structure perform the following:

- View – Shows the interface that the person sees (usually, a webpage). The view additives provide records to the user and moves to the Controller for manipulating data.
- Model – Defines the statistics for the software [typically, the facts is saved in a database (DB)].
- Controller - The controller provides the interface between the View and the model.

# 6. PROJECT STRUCTURE

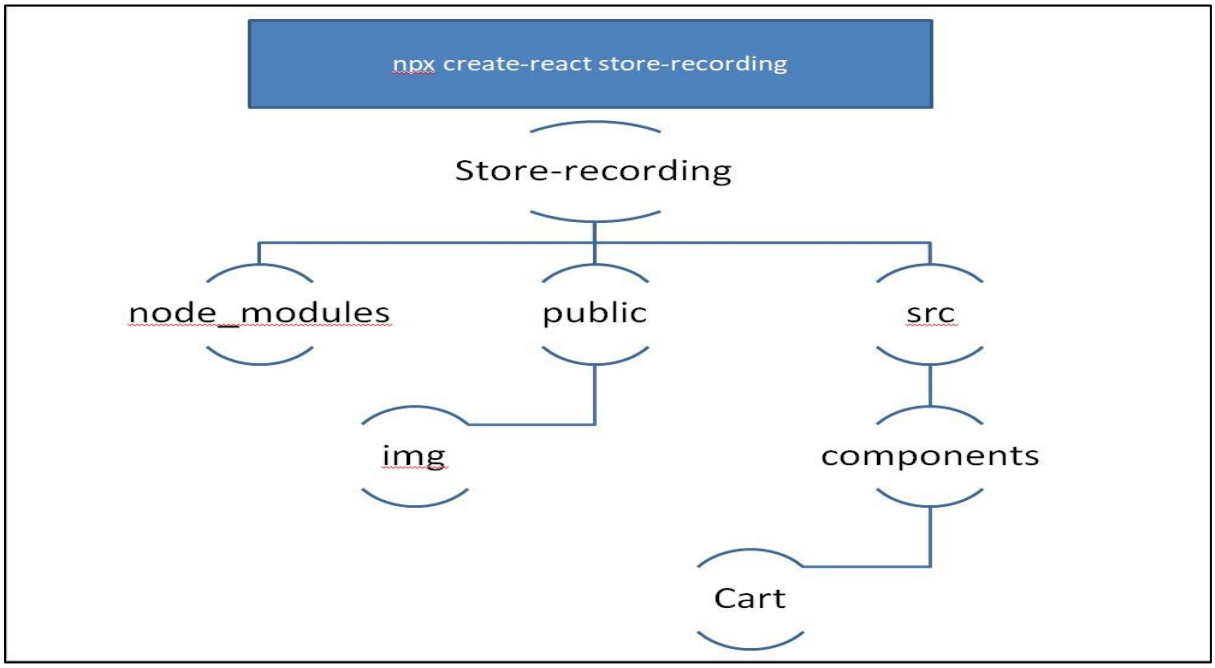


**Figure 2: Structure of store-recording website.**

A project structure provides the framework within which the development effort will be accomplished.

# 7. CUSTOMER INTERFACE LAYOUT

The customer can browse a product, add a product to the cart, change personal information, check shopping history and checkout or log out. The User, on the other hand, can only browse and add a product to cart. The homepage or interface is the index page of the shop so can be accessed when the address is typed into a browser. The webpage has products images, names, prices, product categories and product brands.  A picture of the homepage is shown in Figure 3.

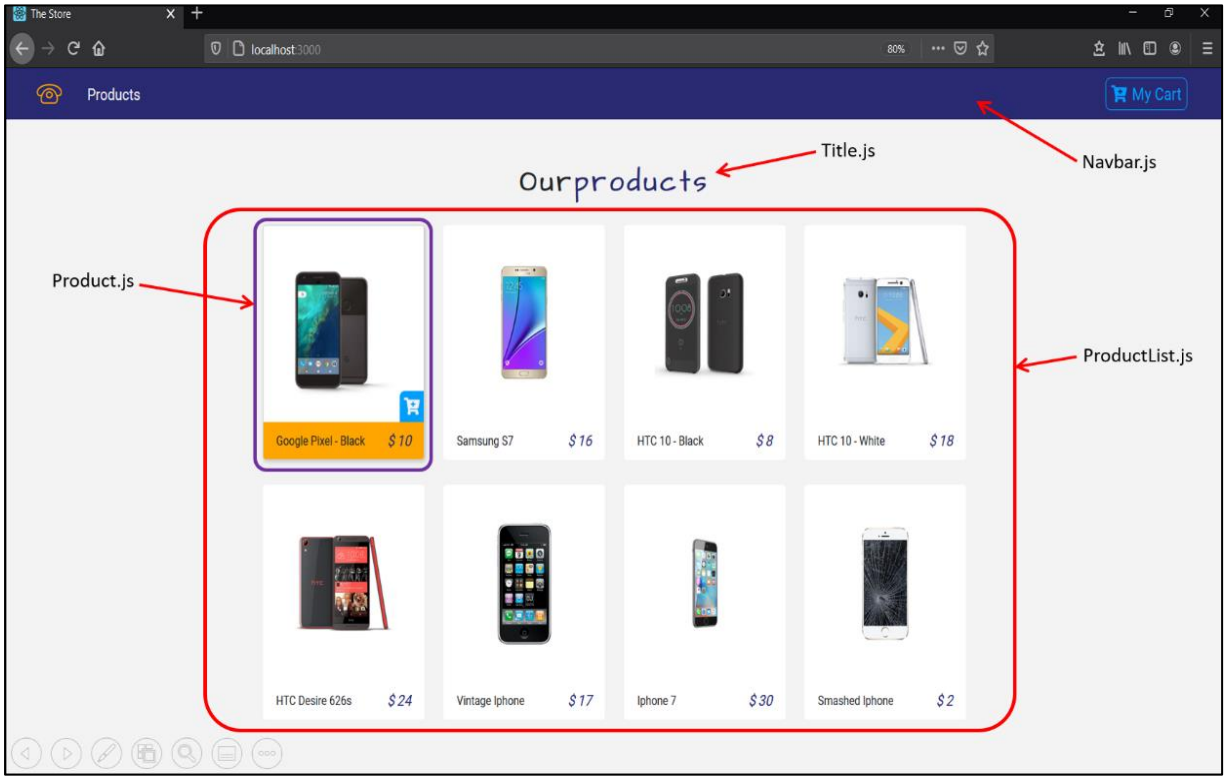


**Figure 3: The homepage of the shop.**

## Title.js

```
import React from 'react'
export default function Title({ name, title }) {
return (
<div className="row">
<div className="col-10 mx-auto my-2 text-center
text-title">
 <h1 className="text-capitalize font-weight-bold">
{name} <strong className="text-blue">{title}
</strong>
</h1>
</div>
</div>
);
}
```

## Navbar.js

```
import React, {Component} from "react"
import {Link} from "react-router-dom";
import logo from "../logo.svg";
```

```
import styled from 'styled-components';
import {ButtonContainer} from "./Button";

export default class Navbar extends Component{

render(){

return(

<NavWrapper className="navbar navbar-expand-sm bg-primary

navbar-dark px-sm-5">

{/*

https://www.iconfinder.com/icons/1243689/call_phone_icon

Creative Commons (Attribution 3.0 Unported);

https://www.iconfinder.com/Makoto_msk */}

<Link to="/">

<img src={logo} alt="store"

className="navbar-brand" />

</Link>

  <ul className="navbar-nav align-items-center">

<li className="nav-item ml-5">

<Link to="/" className="nav-link">

Products

</Link>

</li>

</ul>

<Link to="/cart" className="ml-auto">
```

```
<ButtonContainer>
<span className="mr-2">
<i className="fas fa-cart-plus" />
</span>
MyCart
</ButtonContainer>
</Link>
</NavWrapper>
);
}
}
const NavWrapper = styled.nav`
background: var(--mainBlue)!important;
.nav-link{
color: var(--mainWhite)!important;
font-size: 1.3rem;
text-transform: capitalize;
}
`;
```

## 7.1. Description of products

A product description is the marketing copy used to describe a product's value proposition to potential customers. Providing key product details is critical if you want the shopper to click “Add to Cart” and differentiate your ecommerce website from the competition.

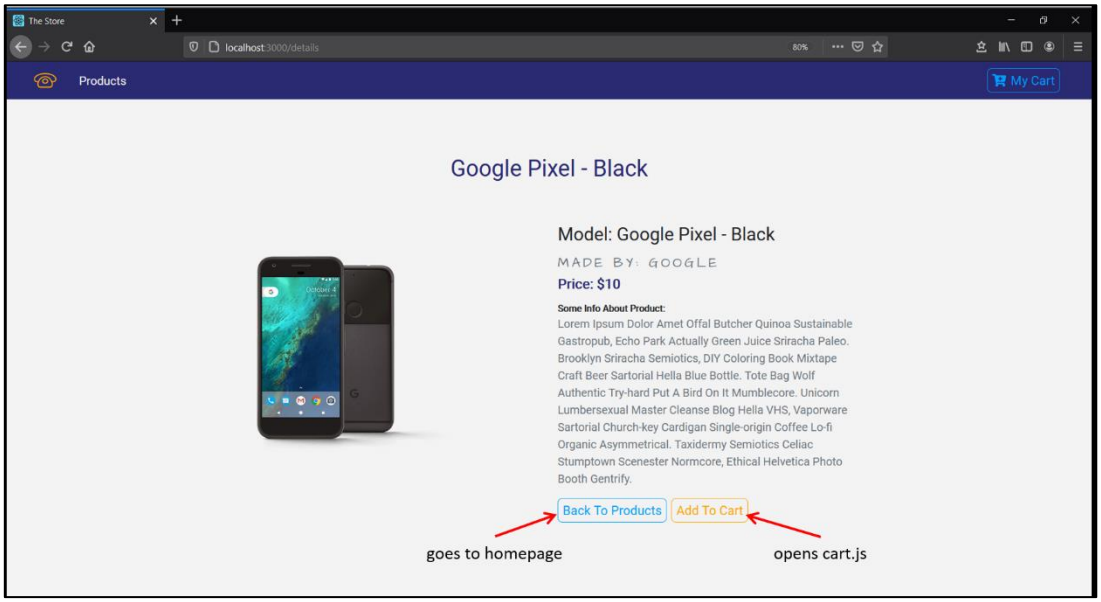


**Figure 4: A brief description of products.**

## Cart.js

```
import React, {Component} from 'react';

import Title from '../Title';

import CartColumns from './CartColumns';

import EmptyCart from './EmptyCart';

import {ProductConsumer} from '../../context';
```

```
import CartList from './CartList'
import CartTotals from './CartTotals';
export default class Cart extends Component{
render(){
return(
<section>
<ProductConsumer>
{value =>{
const {cart} = value;
if(cart.length > 0){
return (
<React.Fragment>
<Title name="your" title="cart" />
<CartColumns />
<CartList value = {value} />
<CartTotals value ={value} />
</React.Fragment>
);
}else {
return <EmptyCart />;
}
}}
```

</ProductConsumer>

</section>

);

}

}

## 7.2. Opening Modal

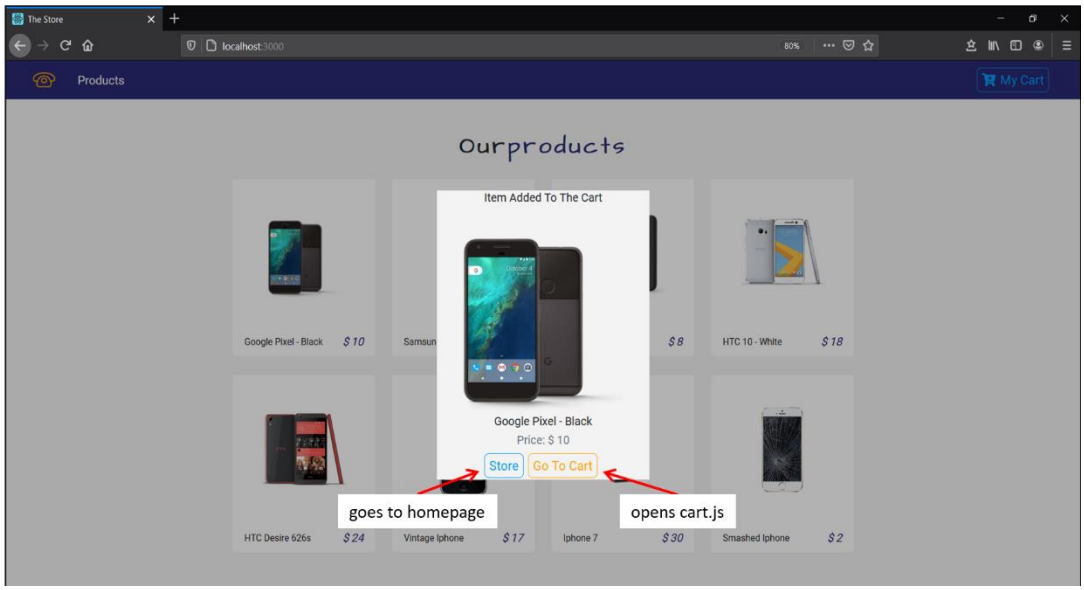


**Figure 5: Modal of our project.**

## Modal.js

import React, { Component } from 'react'

import styled from "styled-components";

import { ProductConsumer } from "../context";

import { ButtonContainer } from './Button';

import { Link } from 'react-router-dom';

```
export default class Modal extends Component {
render() {
return (
<ProductConsumer>
{(value) => {
const { modalOpen, closeModal } = value;
const { img, title, price } = value.modalProduct;
if (!modalOpen) {
return null;
}
else {
return (
<ModalContainer>
<div className="container">
<div className="row">
<div id="modal" className="col-8 mx-auto col-md-6 col-lg-4 text-center text-capitalize p-5">
<h5>Item added to cart</h5>
<img src={img} className="img-fluid" alt="product"/>
<h5>{title}</h5>
<h5 className="text-muted">price :$ {price} </h5>
<Link to='/'>
```

```
<ButtonContainer onClick={() => closeModal()}>
Store
</ButtonContainer>
</Link>
<Link to="/cart">
<ButtonContainer  cart onClick={() =>
closeModal()}>
Go to Cart
</ButtonContainer>
  </Link>
</div>
</div>
</div>
</ModalContainer>
);
}
return
}
}
</ProductConsumer>
 )
}
```

```
}

const ModalContainer = styled.div`

position: fixed;

top: 0;

left: 0;

right: 0;

bottom;

background: rgba(0,0,0,0.3);

display:flex;

align-items: center;

justify-content: center;

#modal {

background: var(--mainWhite);

}

`;
```

## 7.3. Empty cart

Initially the cart is empty as there is no products been added to the cart. The following picture depicts the Empty cart before products being added to it.

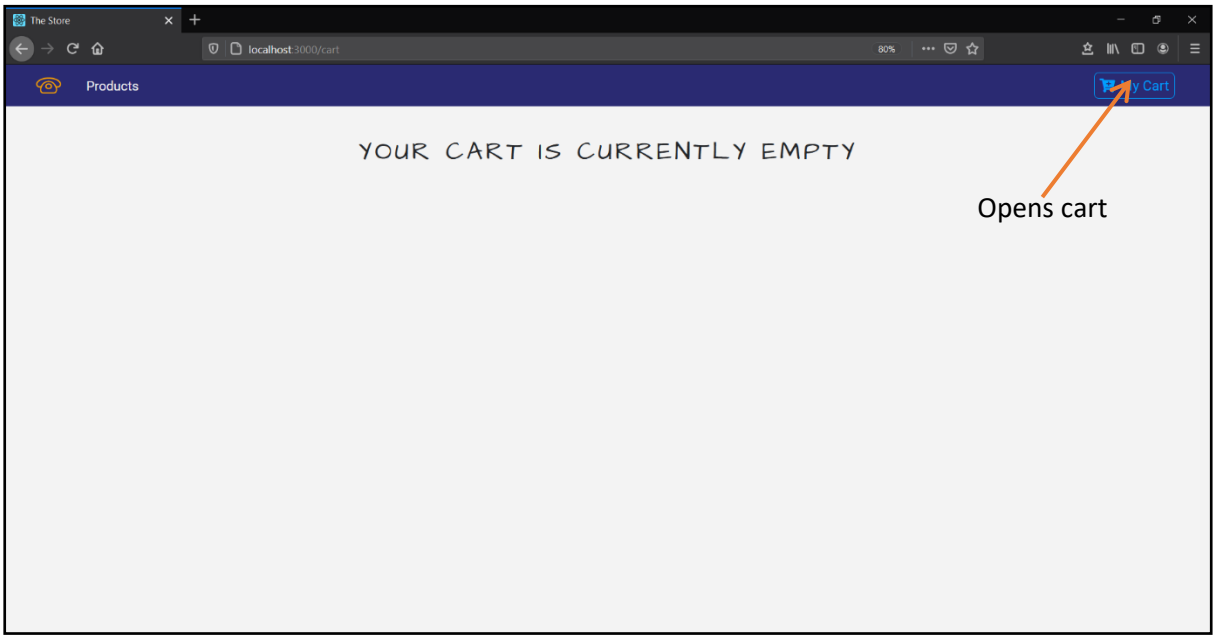


**Figure 6: Empty cart.**

## EmptyCart.js

```
import React from 'react'
export default function EmptyCart() {
return (
<div className = "container mt-5">
<div className = "row">
<div className = "col-10 max-auto text-center text-title">
<h1>Cart is currently empty!</h1>
</div>
</div>
</div>
 ) }
```

## 7.4. Shopping Cart

The customer can add products to the shopping cart. They can also update the shopping cart from time to time until they are done shopping. They can even remove or clear the cart using the clear cart button and the customers can view the total amount of the product after the tax amount is being added.

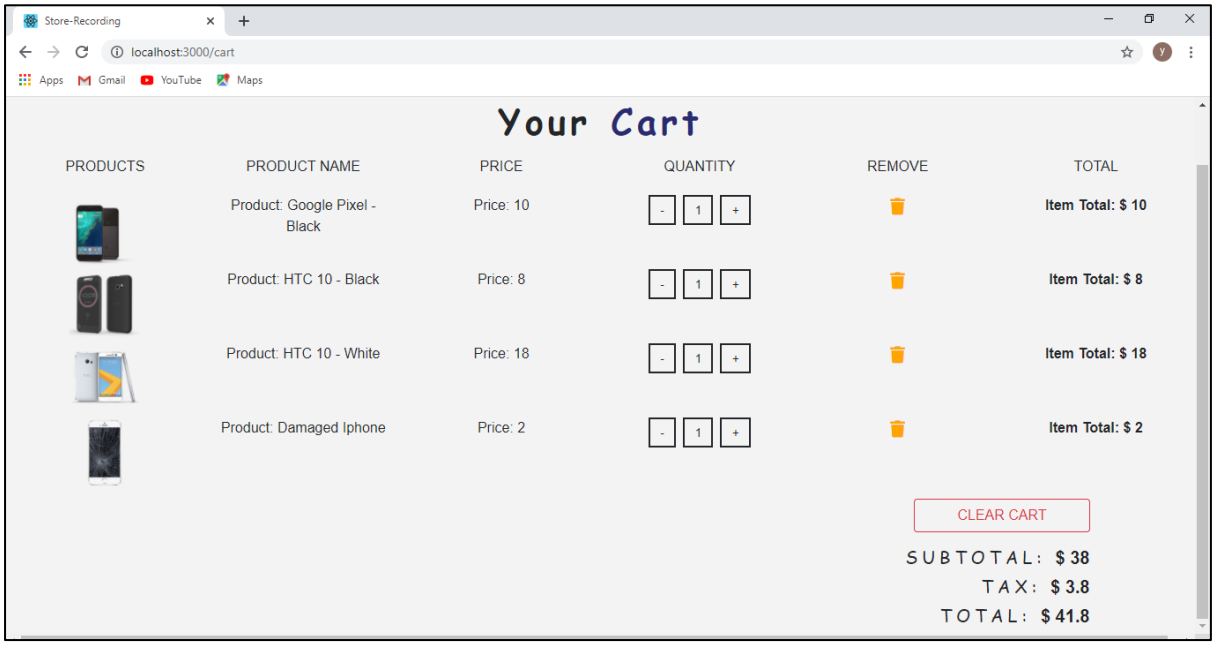


**Figure 7: Shopping Cart.**

## CartList.js

```
import React from 'react'

import CartItem from './CartItem';

export default function CartList({ value }) {

const {cart} = value;

console.log(value, cart);
```

```
return (
<div className = "container-fluid">
{cart.map(item =>{
return <CartItem key = {item.id} item = {item}
value = {value} />;
})}
</div>
);
}
```

CartItem.js

```
import React from 'react'
export default function CartItem({item, value}) {
const {id, title, img, price, total, count} = item;
const {increment, decrement, removeItem} = value;
return (
<div className = "row my-1 text-capitalize text-center">
<div className = "col-10 max-auto col-lg-2">
<img
src = {img}
```

```
style = {{width: "5rem", height: "5rem"}}

  className = "img-fluid"

alt = "product"

/>

</div>

<div className = "col-10 max-auto col-lg-2">

<span className = "d-lg-alone">product: </span>

{title}

</div>

<div className = "col-10 max-auto col-lg-2">

<span className = "d-lg-alone">price: </span>

{price}

</div>

<div className = "col-10 max-auto col-lg-2 my-2 my-lg-0">

<div className = "d-flex justify-content-center">

<div>

<span className = "btn btn-black mx-1" onClick={() =>decrement(id)}

>-</span>

<span className = "btn btn-black mx-1">{count}</span>
```

```
<span className =  "btn btn-black mx-1" onClick={() =>increment(id)}>

+

</span>

</div>

</div>

</div>

{/* */}

<div className = "col-10 mx-auto col-lg-2">

<div className = "cart-icon" onClick={() =>removeItem(id)}>

 <i className = "fas fa-trash"/>

</div>

</div>

<div className = "col-10 mx-auto col-lg-2">

<strong>item total: $ {total}</strong>

</div>

</div>

);

}
```

## CartTotal.js

```
import React from 'react';
import {Link} from 'react-router-dom'
export default function CartTotals({ value }) {
const{cartSubTotal, cartTax, cartTotal, clearcart}= value;
return( <React.Fragment>
<div className = "container">
<div className = "row">
<div className = "col-10 mt-2 ml-sm-5 ml-md-auto col-sm-8 text-capitalize text-right">
<Link to = '/'>
<button
className = "btn btn-outline-danger text-uppercase mb-3 px-5"
type="button"
onClick= {() => clearcart()}
>
clear cart
</button>
</Link>
<h5>
<span className = "text-title">
subtotal:  </span>
```

```
<strong>$ {cartSubTotal}</strong>
</h5>
<h5>
<span className = "text-title">
tax:  </span>
<strong>$ {cartTax}</strong>
</h5>
<h5>
<span className = "text-title">
total:  </span>
<strong>$ {cartTotal}</strong>
</h5>
</div>
</div>
</div>
</React.Fragment>
    );
    }
```

## 7.5. Error Page:

An error message is a message displayed to the user by an operating system or application when an unexpected condition happens. In most cases, error messages are displayed with the help of dialog boxes by the operating system or application. If an incorrect URL is browsed, the 404 (Page not found) error pops up.

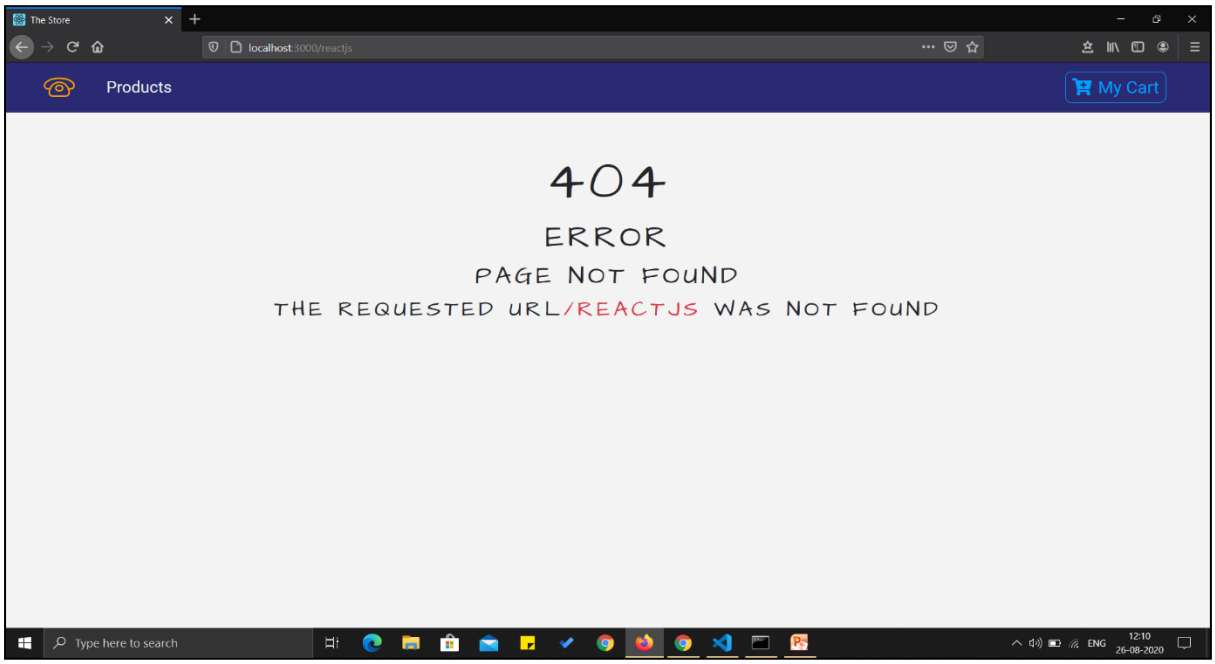


**Figure 8: Error Message (Page not found).**

## Default.js

```
import React, {Component} from 'react';

export default class Default extends Component{

render() {
```

```
console.log(this.props);

return (

<div className="container">

<div className="row">

<div className="col-10 mx-auto text-center text-title text-uppercase pt-5">

<h1 className="display-3">404</h1>

<h1>error</h1>

<h2>page not found</h2>

  <h3>

the requested URL{" "}

<span className="text-danger">"{this.props.location.pathname}"</span> was

not found

</h3>

</div>

</div>

</div>

);

}

}
```

# 8. CONCLUSION

E-commerce is continuously progressing and is becoming more and more important to businesses as technology continues to advance and is something that should be taken advantage of and implemented. From the inception of the Internet and e-commerce, the possibilities have become endless for both businesses and consumers. Creating more opportunities for profit and advancements for businesses, while creating more options for consumers. However, just like anything else, e-commerce has its disadvantages including consumer uncertainties, but nothing that cannot be resolved or avoided by good decision-making and business practices. The electronic shop was developed using ReactJs framework. Any consumer can browse products, add, replace, or delete a product from the cart.