David Zeff 207875147 Maha Elriati 208313197

React Full Stack Final Project

Instructions:

navigate to Server file, then when inside the server file type “npm start” make sure to install all the dependencies first using “npm i”, the server should be running on locally now.

navigate to Client/my-app file, then when inside the my-app file type “npm start” make sure to install all the dependencies first using “npm i”, the frontend should be running locally.

Note: it might take over 10 seconds to fetch the products from database so if you see nothing at first, it’s probably because the products are being fetched.

Components:

Home component:

our “home” component implements the homepage of the website, its has 3 states : iscartopen, this state is what controls the toggle of the cart icon and will display the cart container if it is open, the second state is the cart list state, this will hold all the items in the cart, the third state is the products state , it will be initialized to empty and when we fetch the products from server the products state will change depending on if we manage to retrieve the products from server or not.

```
const Home = () => {
  //for the cart toggle
  const [isCartOpen, setIsCartOpen] = useState(false);
  //for handling the products in cart
  const [cartList, setCartList] = useState([]);
  //for the products that will be fetched from db
  const [products, setProducts] = useState([]);
  //
```

We will expand on functionality and how each state affects the page later.

To implement the items remains in cart even after leaving the page we store them in local storage and retrieve them when the page is loaded, to help us we utilize useeffect for this purpose

```
// this is so that when we leave page and return the items will sti
useEffect(() => {
  const storedCartItems = localStorage.getItem("cartItems");
  //if the items are not empty set cart to have the items
  if (storedCartItems) {
    setCartList(JSON.parse(storedCartItems));
  }
}, []);

useEffect(() => {
  localStorage.setItem("cartItems", JSON.stringify(cartList));
}, [cartList]);
```

When the cart icon is clicked we have a function that will be triggered which will set the cart state to true which will display the cart

```
const toggleCart = () => {
  setIsCartOpen(!isCartOpen);
};
```

Each card has a addtocart button that when clicked will trigger a add to cart function which will take the product clicked on as a parameter

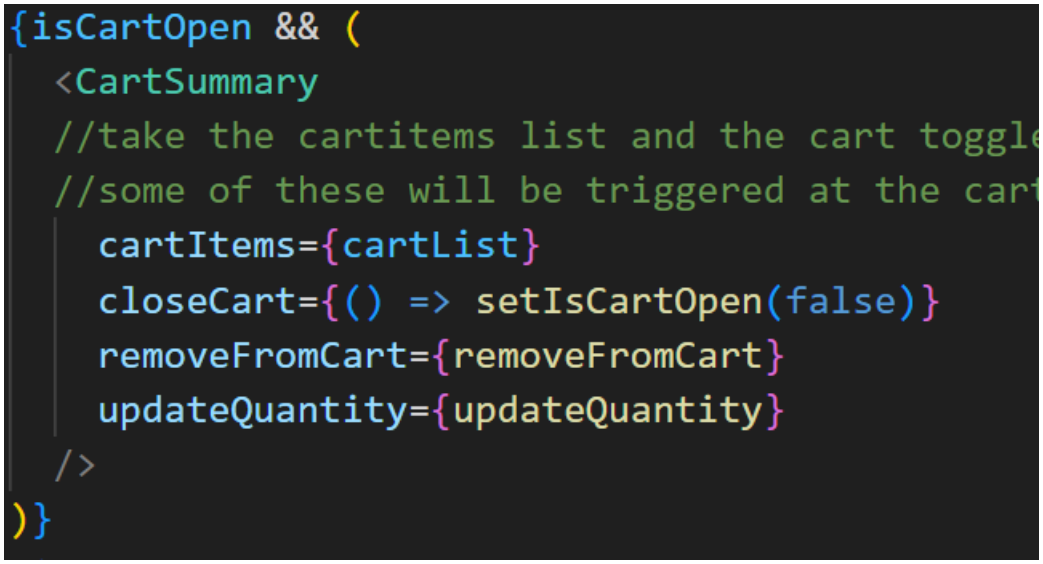
```
const addToCart = (newItem) => {
  let itemExists = false;
  let updatedCartList = [];
```

first, we will initialize a empty array which will hold the updated cart, we will check if item exists, if it does then we will add a modified version to the cart, if not we will add the product to the cart, all older products that were already in cart will be added to the cart.

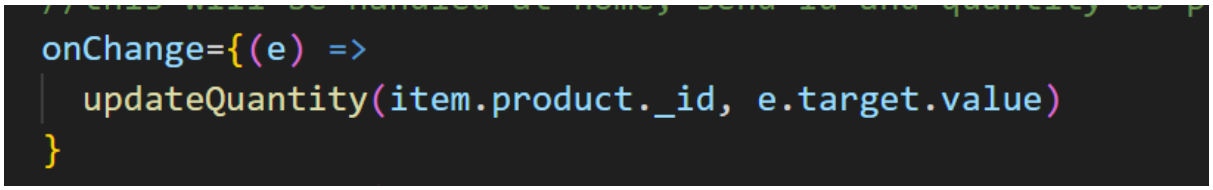
We have a similar function that removes an item to cart it, it will initialize an empty cart and push all items that don't match the product we want to remove and then update the state of the cart to be the updated cart

```
const removeFromCart = (productId) => {
  const updatedCartList = [];
  //initialize empty cart that will have the up
  for (let i = 0; i < cartList.length; i++) {
    //all items that dont match the one we are
    if (cartList[i].product._id !== productId)
      updatedCartList.push(cartList[i]);
    }
  }
  //set the updated cart
  setCartList(updatedCartList);
};
```

We have another function for updating the quantity of each item within the cart. It takes the product id and the quantity and will find the product a=in cart and update it to the new quantity, this function will actually not be triggered within the home component but within the cart summary component, and we have a few other scenarios where functions that are not "local" to the component will be triggered along with send params and then this function which was triggered will be activated within another component. For example, with remove from cart function we send it as a prop to a "cartsummary" component, this will enable us to trigger the function within the other component along with sending parameters if we wish



As we can see the carsummary component is within the home component and has a few props being sent along with removefromcart and updatequantity both are functions which are not local to the cartsummary component but will be triggered and given parameters in that component which will be sent “back” to the function which is within the home component.



This is an element within the cartsummary component and will be triggered any time it changes and will return id and value as parameters.

That’s an example of how this how we used “callback props” in our project.

Anyway, to continue with our home component, we need to fetch products from server to do this we utilize the useeffect and fetch, and make a get request to our server address,

```
// Fetch products
useEffect(() => {
  //api get address on server
  fetch("http://localhost:3001/products")
    .then((response) => {
      if (!response.ok) {
        alert("error" + response.statusText);
      }
      return response.json();
    })
    .then((data) => {
      //if no error then set products with the data ret
      setProducts(data);
    })
```

If the request is successful, we will set the products to be the ones we just fetched.

The home compenet has a navbar component which is visible and has a home icon that is a react icon along with a cart icon which is also imported from react icons, it also has a cartSummary component which is only visible when the cart icon is clicked, and also each product will be sent as a prop to a card component and the cards will be displayed in a 3 by 3 grid, every single component has its own css file that is used for styling.

```
{/*nav bar component that takes the togglecart function for th
  <Navbar toggleCart={toggleCart} cartItems={cartList} />
  <div className="homeContainer">
    {/*logo img from thats imported from a file within the pro
```

The above shows the navbar component within the home component.

```
{products.map((product) => (
  <Card
    key={product._id}
    product={{
      _id: product._id,
      name: product.name,
      price: product.price,
      image: `images/${product.image}`,
      description: product.description
    }}
    //every card will take the function that will
    addToCart={addToCart}
  />
```

The above we can see the page displaying all the products as card components within the home component.

```
{isCartOpen && (
  <CartSummary
  //take the cartitems list and the cart toggle
  //some of these will be triggered at the cartS
    cartItems={cartList}
    closeCart={() => setIsCartOpen(false)}
    removeFromCart={removeFromCart}
    updateQuantity={updateQuantity}
  />
)}
```

The above we see that the cart component will only be displayed if the cartopen state is true (its set to true/false when we click on the cart icon).

The readme is getting a bit long but for any other info look at the source code which has comments for more info about home component.

CartSummary component:

The cartsummary component is what displayes all the items in cart and also is where we proceed to order, it takes 4 props the first prop cart items is what holds the item in cart info, the second prop is close cart which is a call back function that will close the cartsummary display if triggered, the third prop is a remove from cart call back that if triggered will send as a parameter the id of the item we wish to remove, the function itself is within the home component, the last prop is the quantity updater which if triggered it will update the quantity of the product by sending back to the function the id and quantity which are relevant to update the product.

```
const CartSummary = ({
  //has all the products info
  cartItems,
  //the 3 below will be triggered here bu
  closeCart,
  removeFromCart,
  updateQuantity,
}) => {
```

Within the cart summary component we have the option to proceed to order, when we do this we want to navigate to the order page and send along all the information fo the products which will be in the order, to do this we use the “useNavigate” given by the react router dom, this will allow us to navigate to another rpage along with all the product info we need

```
 const navigate = useNavigate();
/when clicked proceed to order we will navigates to order page witl
 const handleProceedToOrder = () => {
   navigate("/order", { state: { cartItems: cartItems } });
 };
```

We have a routing logic set up where each component has its own route

In our App.js

```
function App() {
  return (
    <Router>
      <div className="App">
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/Home" element={<Home />} />
          <Route path="/order" element={<OrderPage />} />
        </Routes>
      </div>
    </Router>
  );
}
```

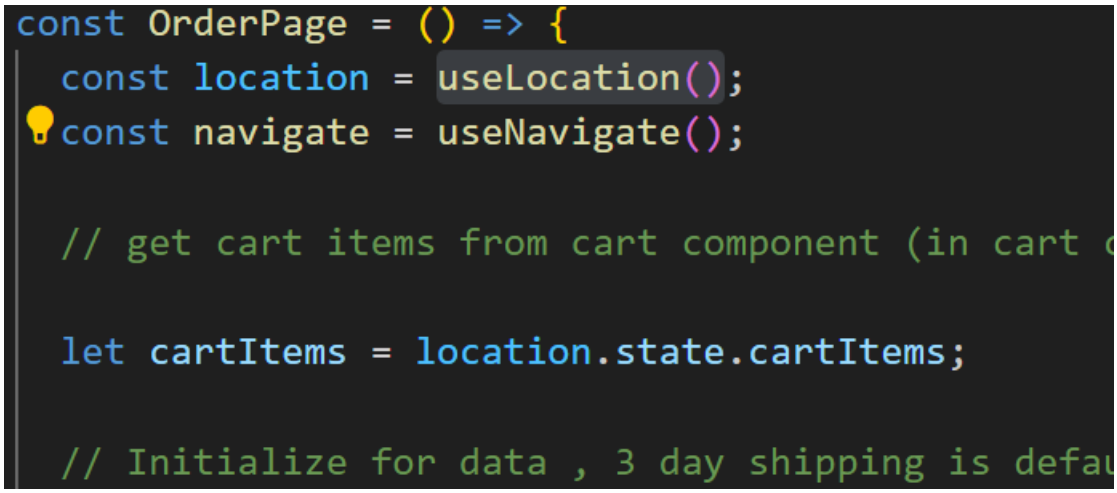
In our cartsummary component we will display all items in cart by mapping the caritems.

```
cartItems.map((item) => (
  <div key={item.product._id} className="cartItem">
    <img
      src={item.product.image}
      alt={item.product.name}
      className="cartItemImg"
    />
    <div className="cartItemDetails">
      <h3 className="cartItemTitle">{item.product.name}</h3>
      <p className="cartItemText">
        Price per Unit: ${item.product.price}
      </p>
      <p className="cartItemText">
```

If we click on the proceed to order we will navigate to the order page which is also a component,and we will send along the cart items info as we saw above.

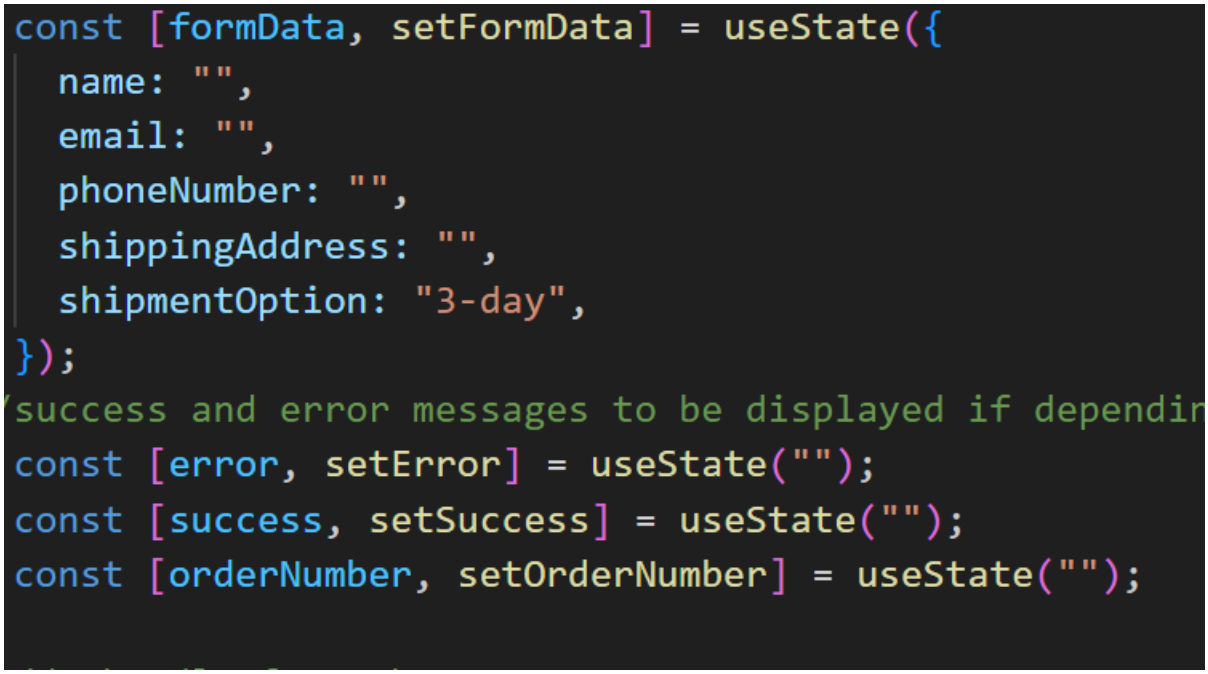
OrderPage component:

The order page component will be the page where we submit our order, it will get the items from useLocation().state which will hold the items which we inserted in the cartsummary component page,



We also use the useNavigae because if we have a successful order we will navigate back to the homepage.

In the order component we have 4 states, 1 for the error message 1 for a success message which will display the success message if order is successful and also a state for the order number which will be displayed if again if there is success and the last state which will hold our form data which will be initialized with default values.



Anytime one of the fields in the form is changed we will update only the field being changed and use the spread operator to layout all the other fields of the form which were not changed

```
const handleInputChange = (e) => {
  const name = e.target.name;
  const value = e.target.value;

  // spread the form data and update the pr
  const newFormData = {
    ...formData,
    [name]: value,
  };

  // update form
  setFormData(newFormData);
};
```

When we submit the form, we will send a post request to our server and send in the payload all the form data

```
fetch("http://localhost:3001/orders", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    //persnal info
    name: formData.name,
    email: formData.email,
    phoneNumber: formData.phoneNumber,
    shippingAddress: formData.shippingAddress,
    shipmentOption: formData.shipmentOption,
    //all products
```

If the request is successful and we managed to insert the order into our db then we will reset the form and the cart, display the order number and after a 4 second delay we will redirect back to homepage

```
if (result.order) {
  setOrderNumber(result.order._id);
  setError("");
  setSuccess(result.message);
  //reset form
  setFormData({
    name: "",
    email: "",
    phoneNumber: "",
    shippingAddress: "",
    shipmentOption: "3-day",
  });
```

```
  // go back to homepage after some delay
  setTimeout(() => {
    navigate("/");
  }, 4000);
}
```

Card component:

We take all the product info as props and have simple styling using css,

We have a addtocart call back function which will be handled at the homecomponent

```
const handleAddToCart = () => {
  addToCart({
    product: product,
    quantity: 1,
    totalCost: product.price,
  });
};
```

```
  return (
    <div className="cardContainer">
      <img src={product.image} alt={product.name} className="cardIm
      <div className="cardBody">
        <h2 className="cardHeading">{product.name}</h2>
        <p className="cardText">{product.description}</p>
        <p className="cardCost">${product.price}</p>
        <button className="addBtn" onClick={handleAddToCart}>
          Add to Cart
        </button>
      </div>
    </div>
  );
```

NavBar component:

The navbar has the logo and 2 react icons the cart and home icons, it has a carttoggle call back function, when cart icon is clicked this function will be triggered and the cart will be displayed in the home component, we also pass the items length that will be displayed in a badge above the cart icon.

```
<a href="/home">
  <FaHome size={20} />
</a>
{/*cart icon will trigger the cart toggle which is handled at home*/}
<button className="cartIcon" onClick={toggleCart}>
  <FaShoppingCart size={20} />
  {/*if there are items then display the amount in cart*/}
  {cartItems.length > 0 && (
    <span className="cartBadge">{cartItems.length}</span>
  )}
```

Server:

In the server we have a post for adding orders to the db

```
app.post("/orders", (req, res) => {
  const newOrder = req.body;
  //add new order to the db
  db.final_David_Zeff_and_Maha_Elriati.insert(newOrder, (err, result) => {
    if (err) {
      res.status(400).json({ error: err});
    } else {
      res.status(201).json({
        order: result,
        message: "Order success!",
      });
    }
  });
```

And we also have a app.get for getting the products from the db

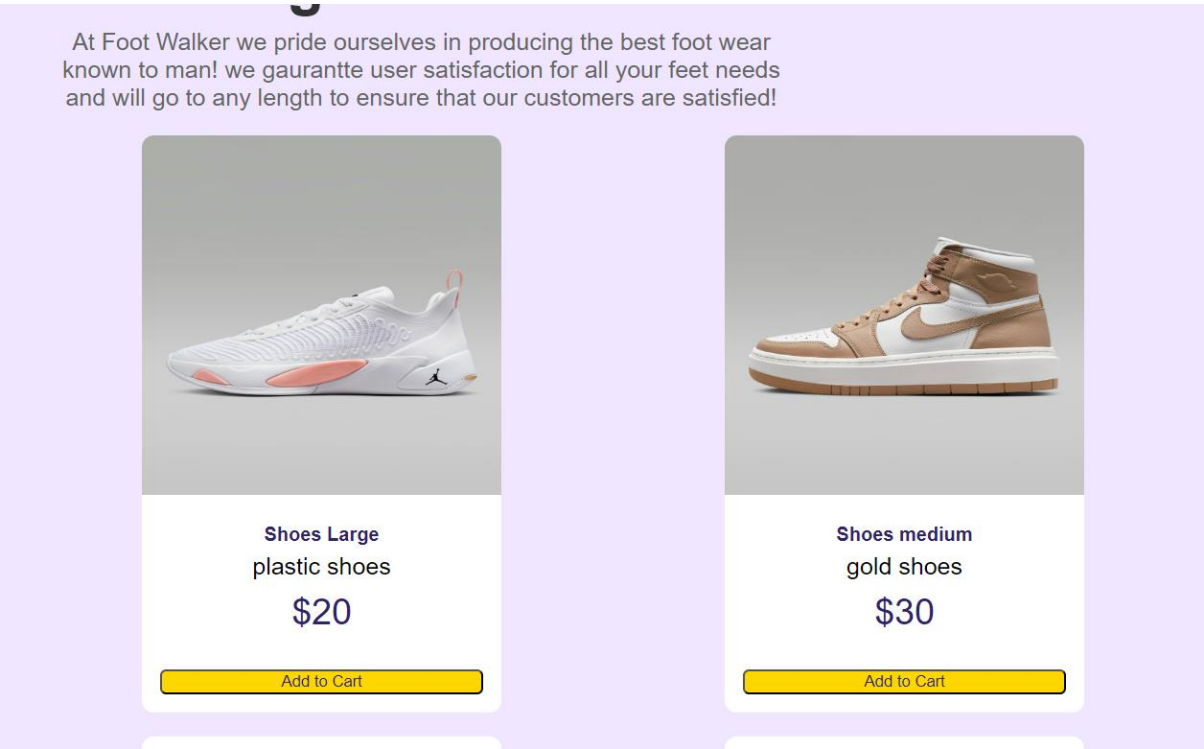
```
app.get("/products", (req, res) => {
  db.products.find({}, (err, products) => {
    if (err) {
      res.status(500).json({ error: "Erorr!" + err });
    } else {
      //if no error return products
      res.json(products);
    }
  });
```

Example:

Foot Walker

Walking for feets sake!

oot Walker we pride ourselves in producing the best foot wear
n to man! we gaurantte user satisfaction for all your feet needs
will go to any length to ensure that our customers are satisfied!

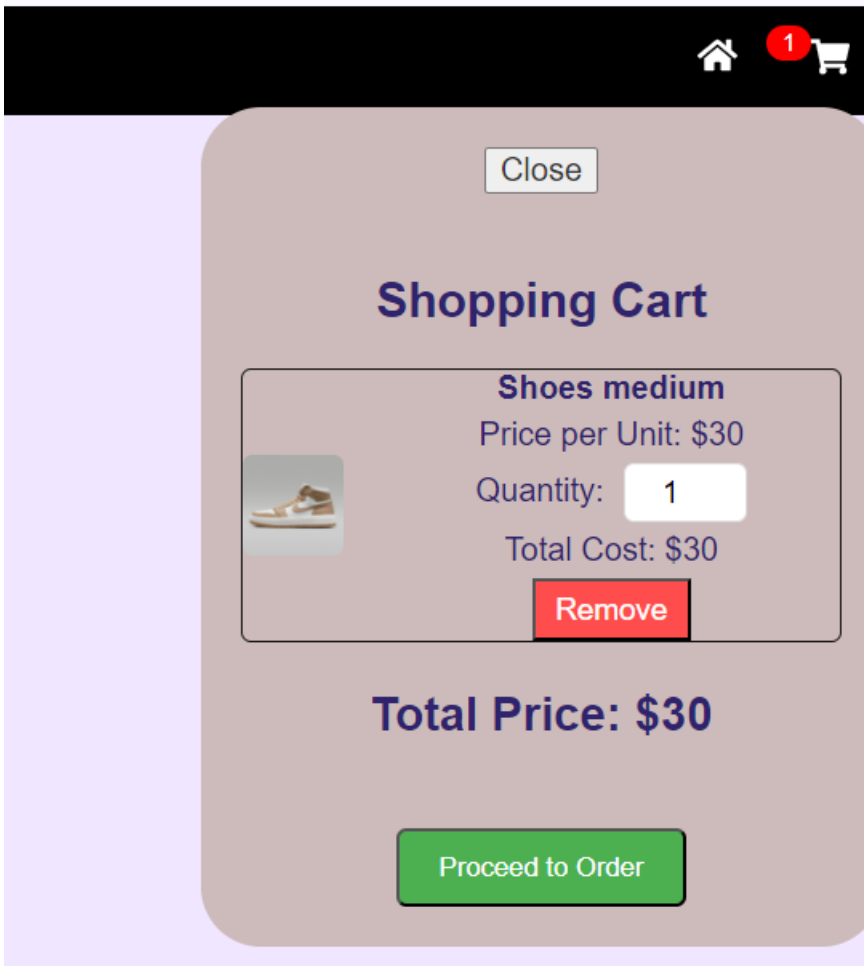


The above screenshots display the homepage, as we can see we have a navbar with the company logo name and to the right we have the home and cart icons, top centre we have the logo picture and the slogan and a description,

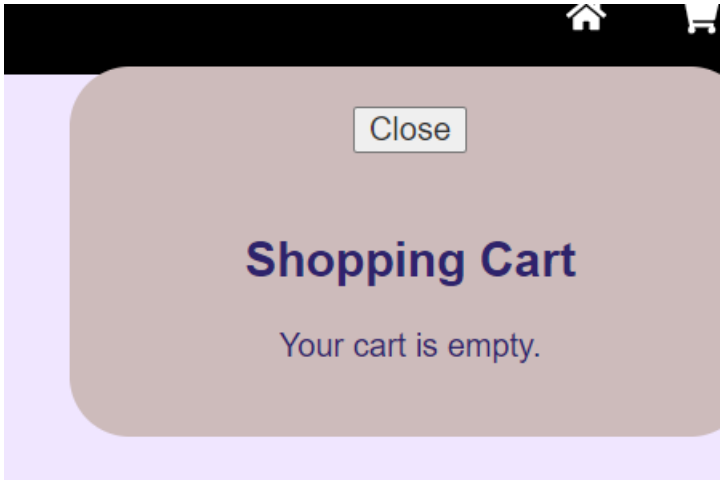
And we can see all the products displayed as cards, thes products we taken from the server.

Lets see what happens when we click on one of the products to add to cart...

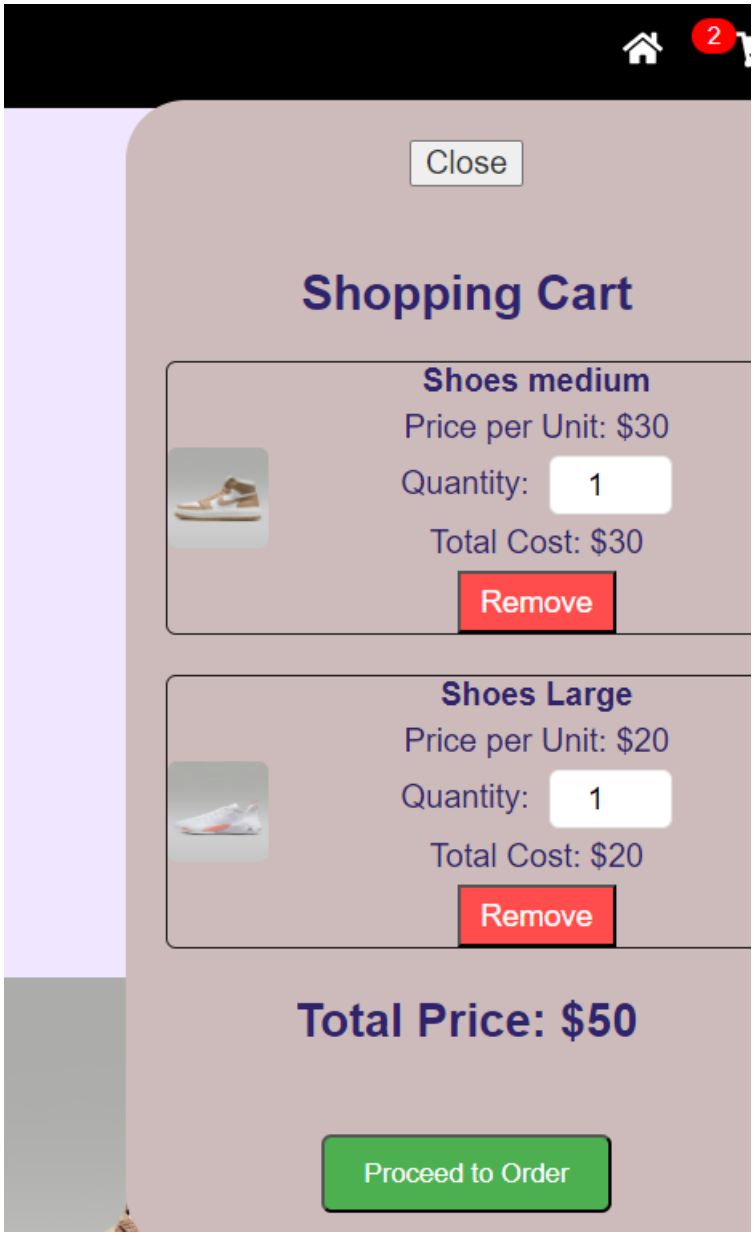
As we can see the badge has increased by 1 because we added a product, lets look inside the cart by clicking on the cart...



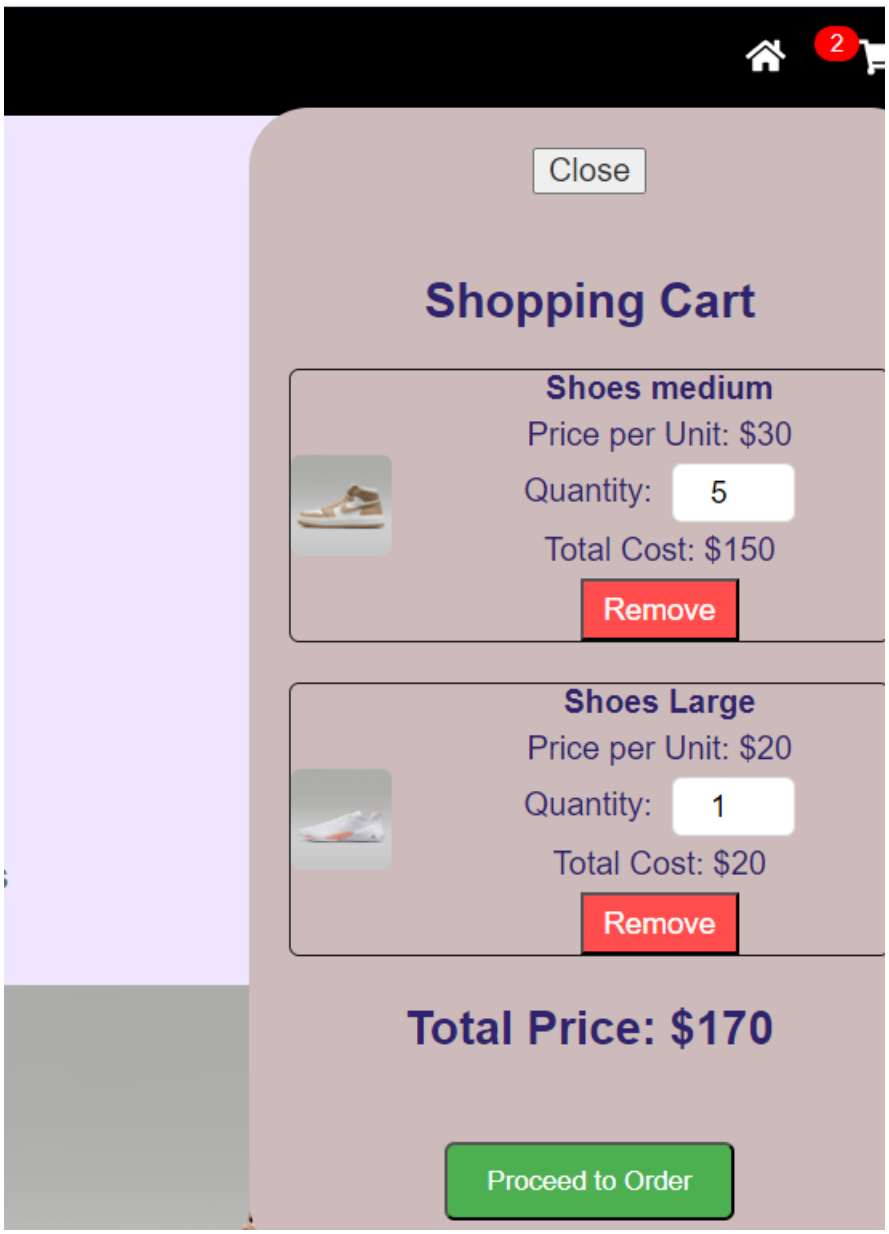
We have the cart displaying the items we added, we have the option to change the quantity and also to proceed to order page and also to remove the item, lets remove the item



Lets add another item



Lets manually adjust the quantity:



We are happy with our selection, lets proceed to order...

Foot Walker

Order Page

Shoes medium

Quantity: 5

Total Cost: $150

Shoes Large

Quantity: 1

Total Cost: $20

Submit Order

Name:

Email:

Total Cost: $20

Submit Order

Name:

Email:

Phone Number:

Shipping Address:

Shipment Option:

3-day Shipment (Fee)

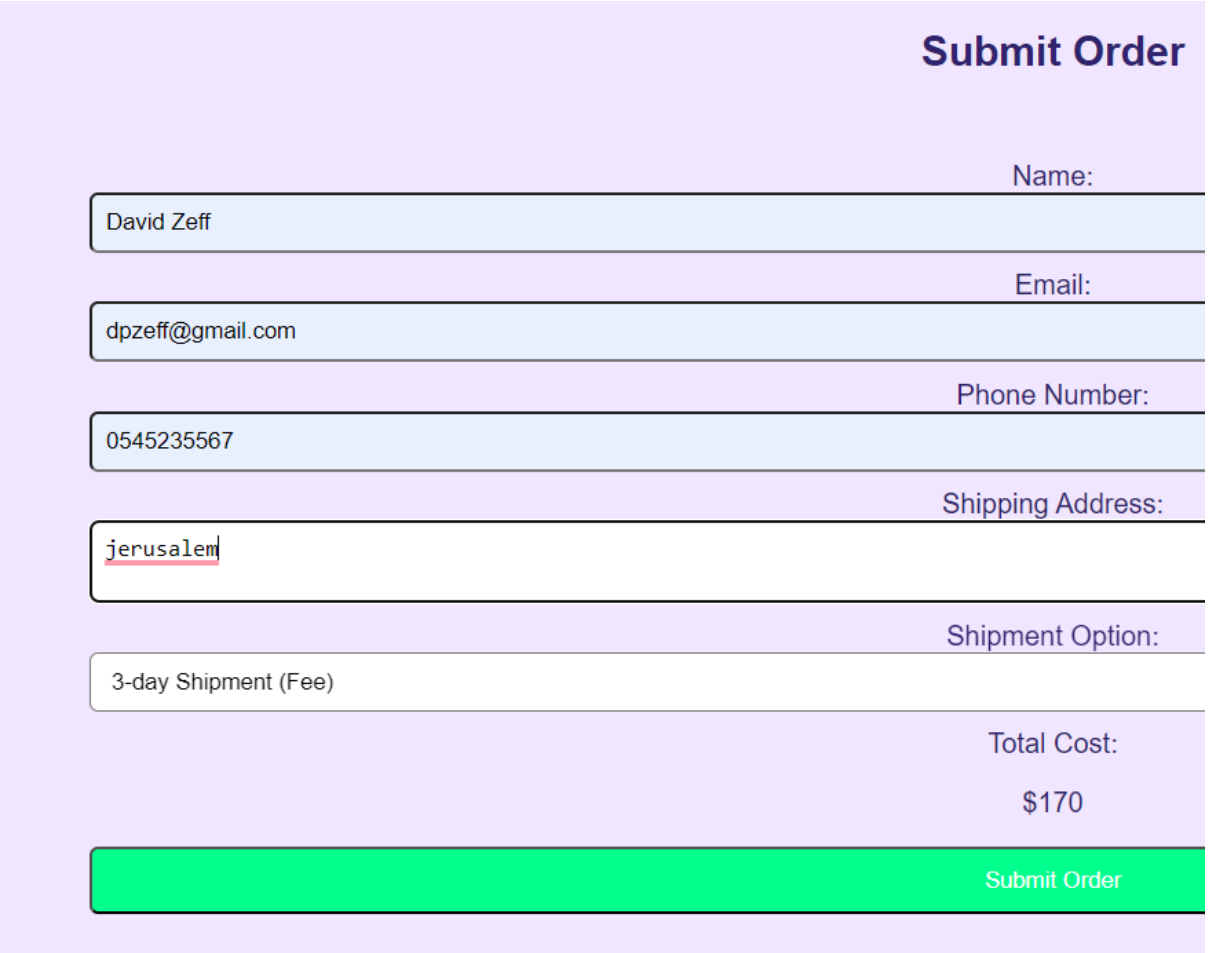
Total Cost:

$170

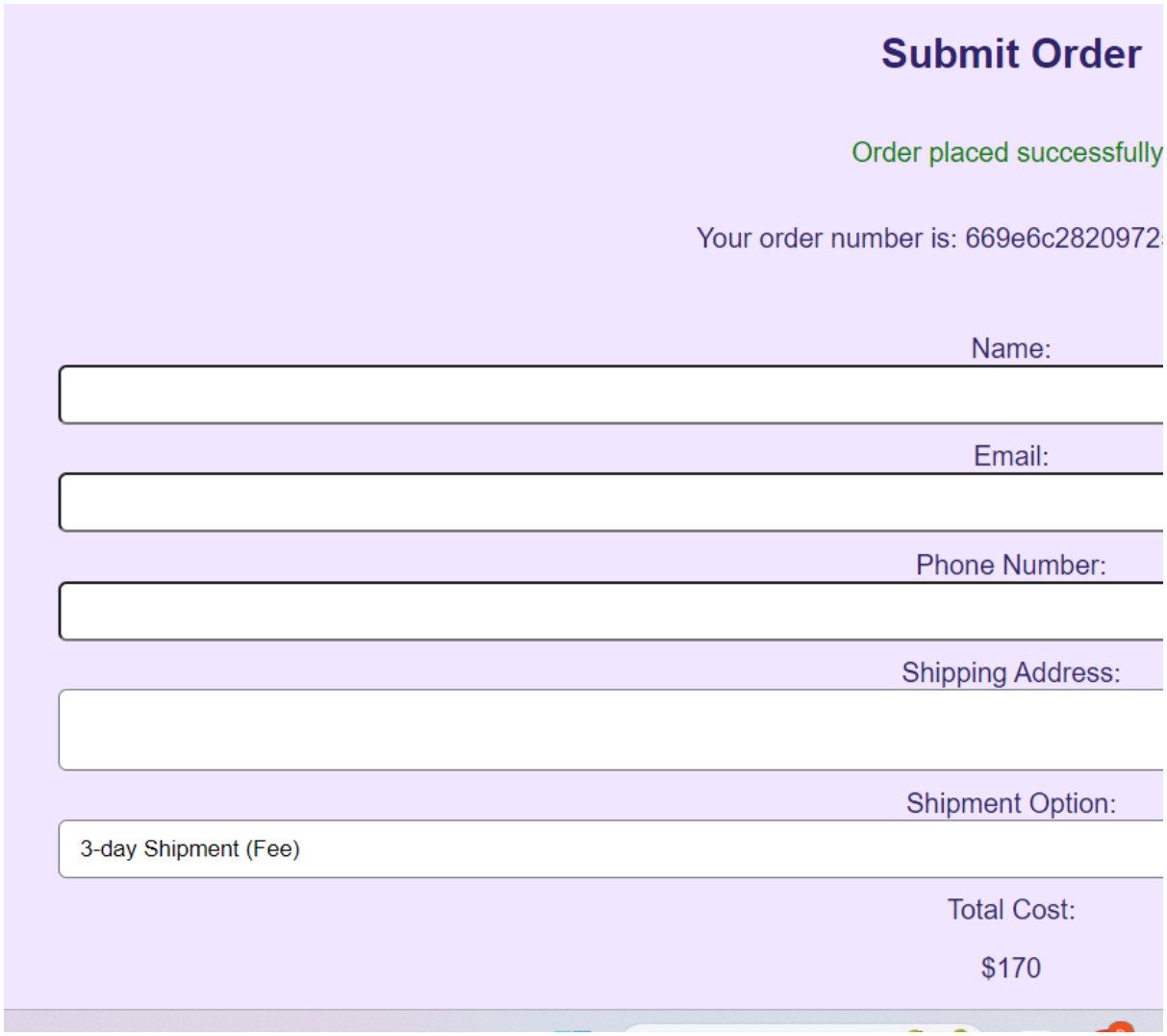
Submit Order

As we can see this is our order page with all the items we selected and a order form where we need to enter our private details

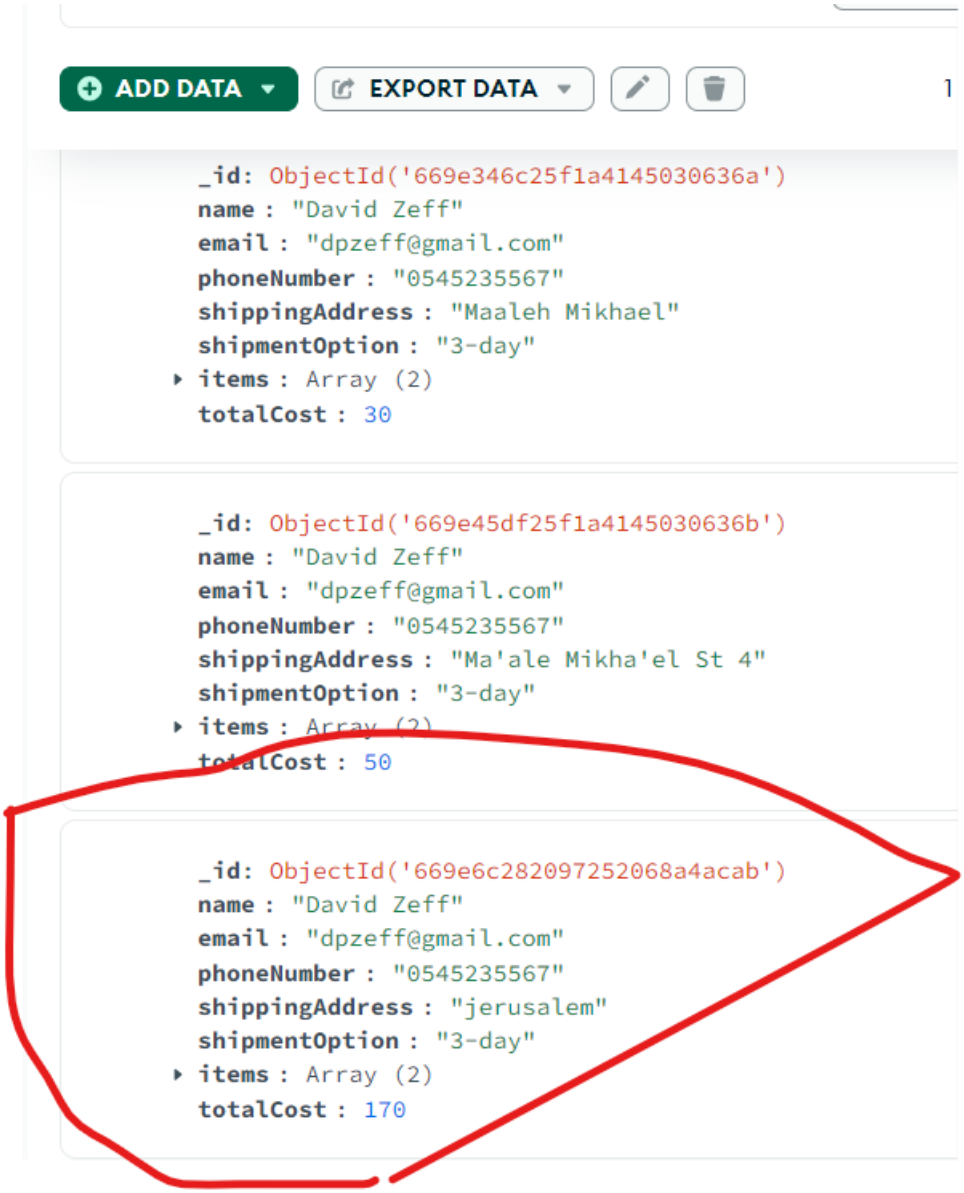
The form will not allow us to submit if we dont fill out all the details,



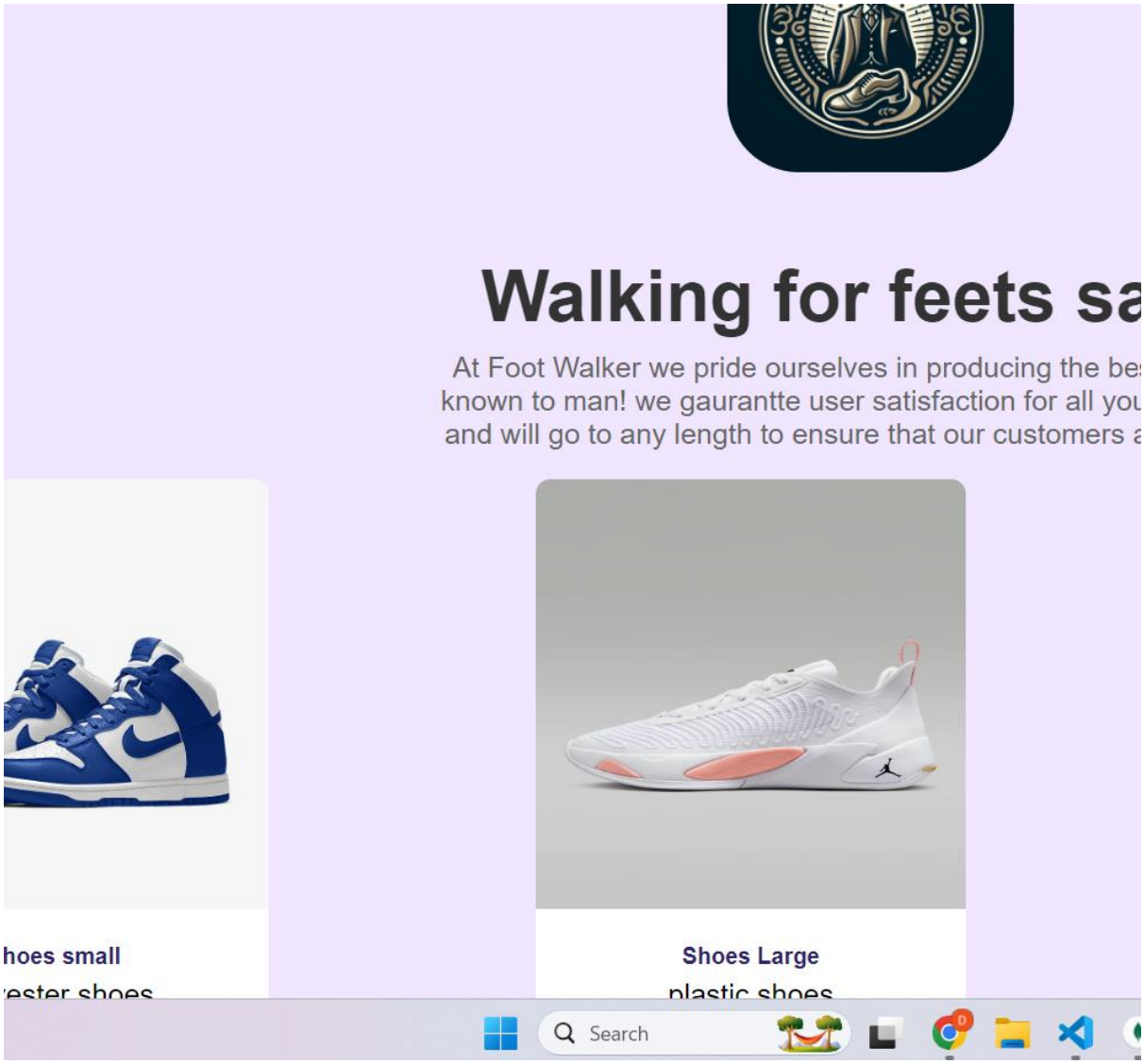
Lets submit the order



He order was successful and we can see a success message along with the order number, lets check the db



As we can see our order has made it to the db



After a few seconds we made it back to the homepage with everything reset and we are ready to make a new order.

For more details check out the source code that has comments.

Thats all, thanks for the awesome course!