SE/ComS319 Construction of User Interfaces, Spring 2023

May 10, 2023

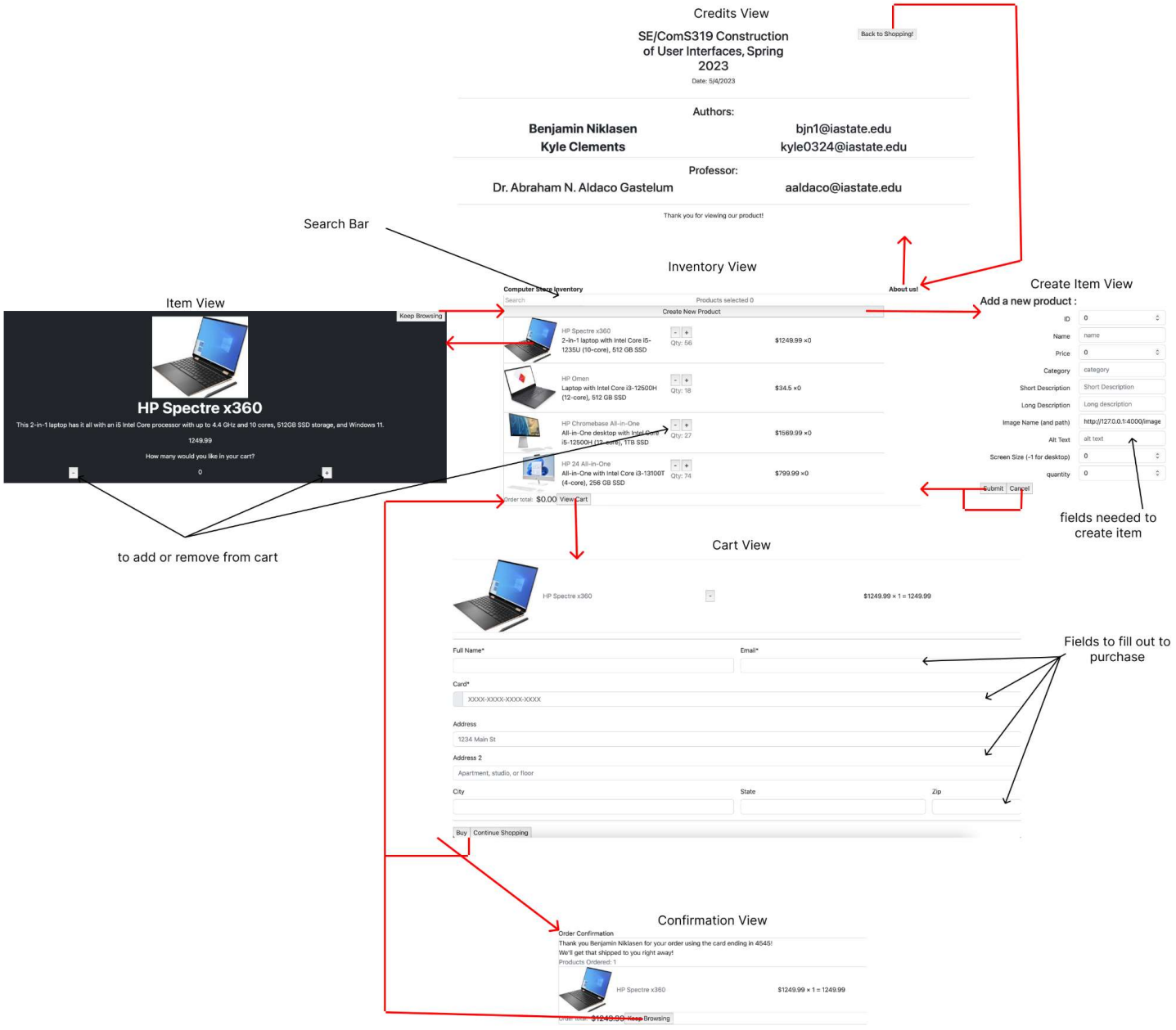
Benjamin Niklasen (bjn1@iastate.edu)

Kyle Clements (kyle0324@iastate.edu)

Professor : Dr. Abraham N. Aldaco Gastelum (aaldaco@iastate.edu)

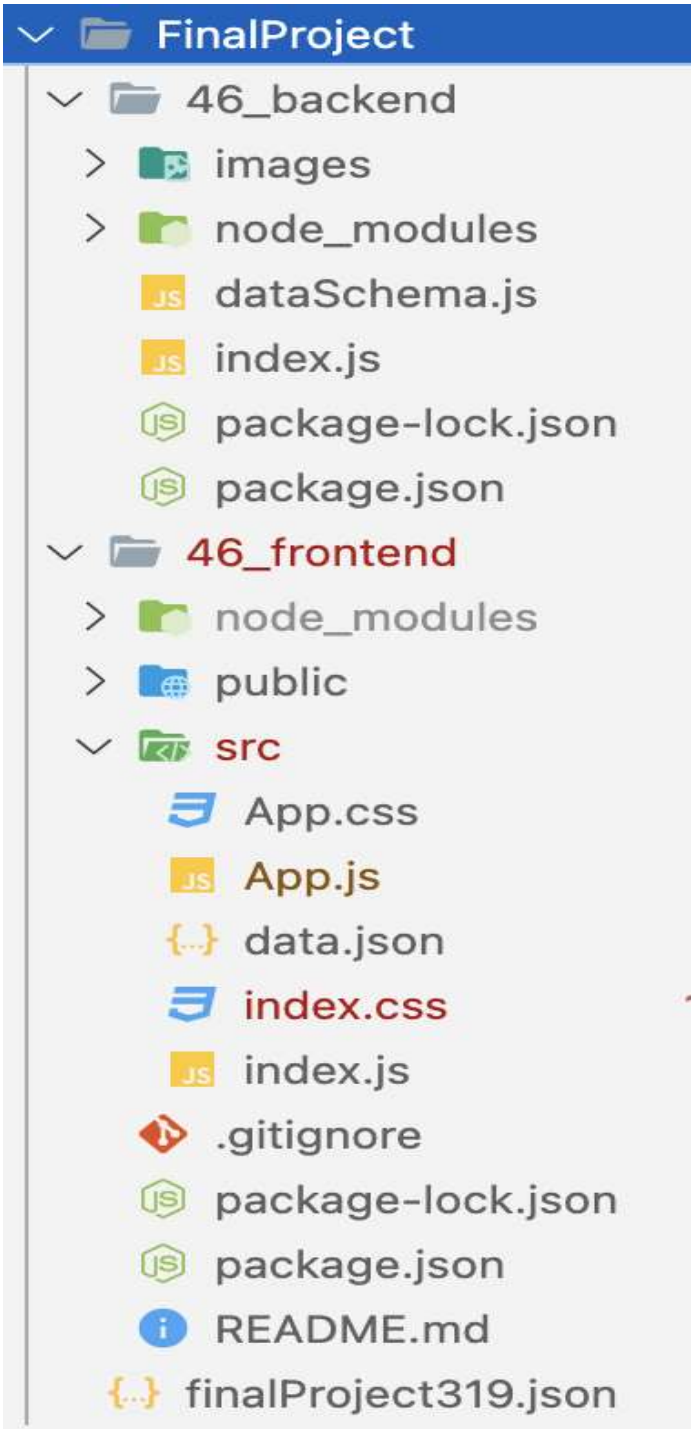
Final Project Documentation

Project Diagram



Description text and graphics of files and directory architecture

This screenshot details the directory of the architecture. The overarching directory is the folder `FinalProject.` Within this directory are two folders, the backend and frontend.



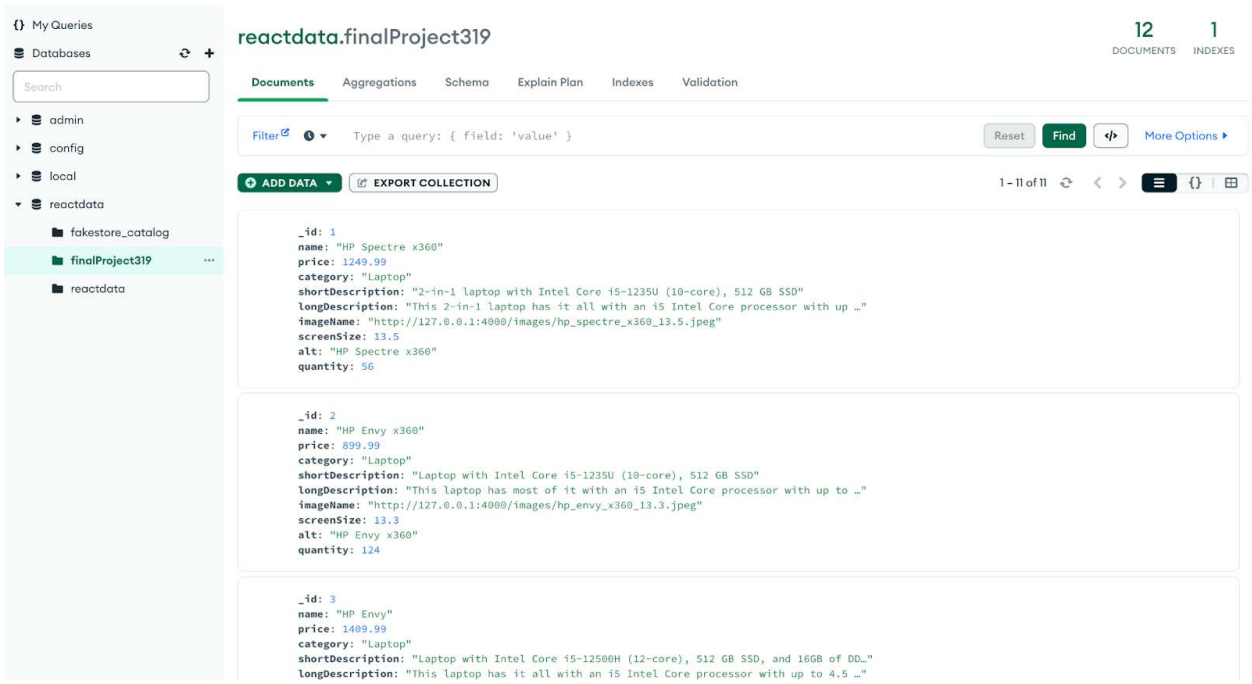
The backend folder contains the necessities for Node JS and Express. The file 'dataSchema.js' contains the schema provided to the Mongoose model to tell the backend what structure to expect for the documents located in the Mongo database.

The file `index.js` contains the code that is run on the backend and handles the requests made by the frontend. Using Express and Node JS, it gives responses to the different calls requested by the frontend. It also provides the images in the `images` folder to be accessible online upon request.

The frontend folder contains the necessities to create a single-page website and all the functionality of the computer catalog. React is used to allow the code to be configured easily using javascript and allows for easy viewing of recently modified changes. It also incorporates bootstrap to make the website look appealing to the average individual.

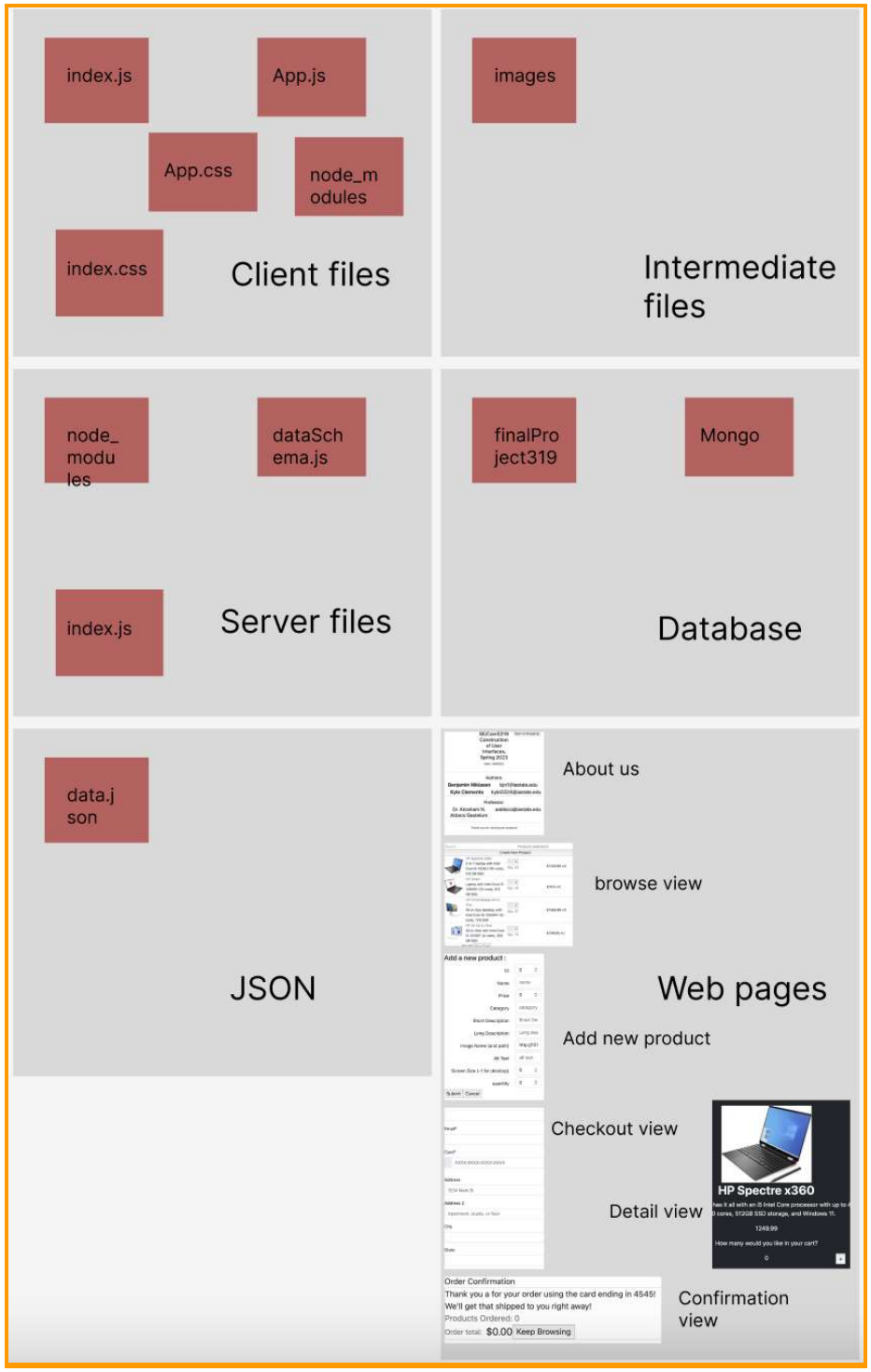
The file `index.js` is the main file but only really contains the base code that responds to React and the call to `App.js`.

The file `App.js` contains the majority of the work and code for the frontend. It is what defines the look and actions the users can take as well as the code that makes the website a single page instead.

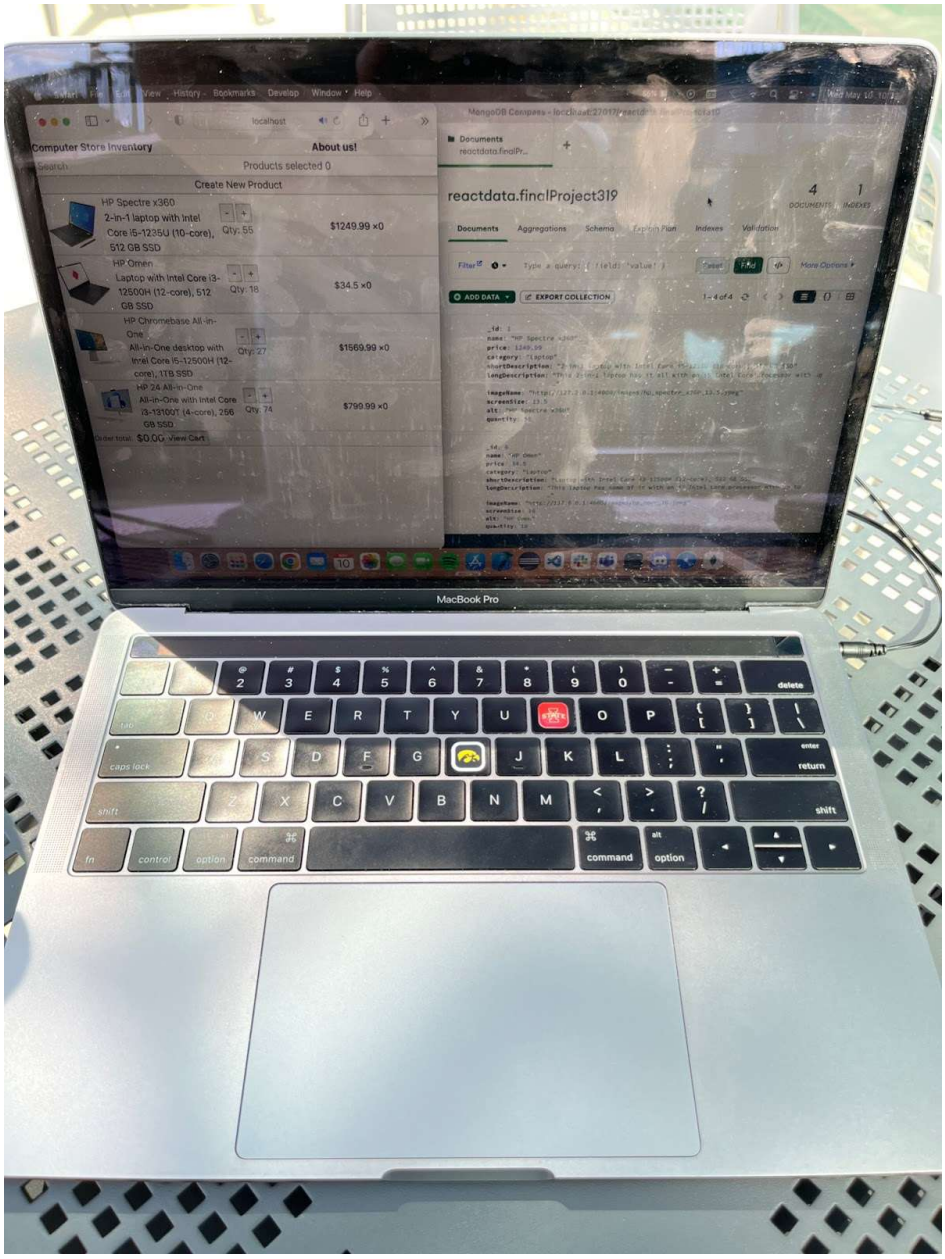


This is what the database looks like. By default (using the default JSON provided) there are 12 entries, detailing products' ids, names, price, category, descriptions, imageName, alternative text, screen size, and quantity of the product available. When an entry is in the database, the only changes that can be made to it (via the website) are either altering / lowering its quantity available (after the user makes a purchase) and deletion (after the user purchases all of a particular product). Entries can be created by the website and are required to have all of the information that it can contain in order to be created.

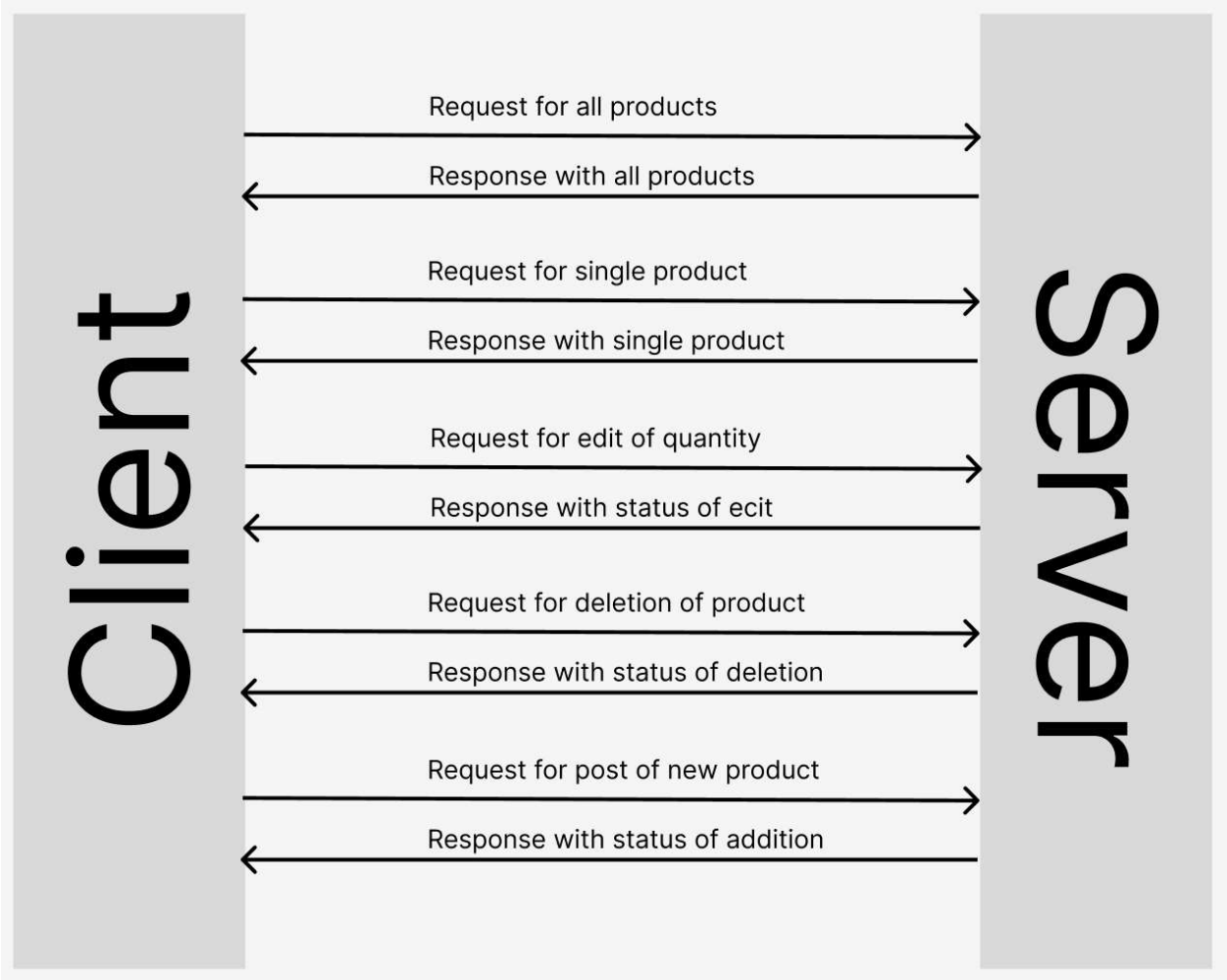
Diagram of the modules that represent user, server, intermediate files, JSON, database, and web pages



Descriptive text and graphics of the client-server architecture



The above picture shows the physical architecture of the webpage. It is being run on a 2016 MacBook Pro using the terminal commands in the software and installation page. The backend and server are also being run locally on it as well. In this document, the client and the server are both the same computer (in this instance) and all communication is local. The web page is being launched using React from the terminal and is running in Safari. The backend is being launched and run by Node in the terminal. The database is being launched and run by Mongo using the terminal and is viewed and maintained by the Mongo Compass application.



This diagram above shows the interactions between clients and the server.

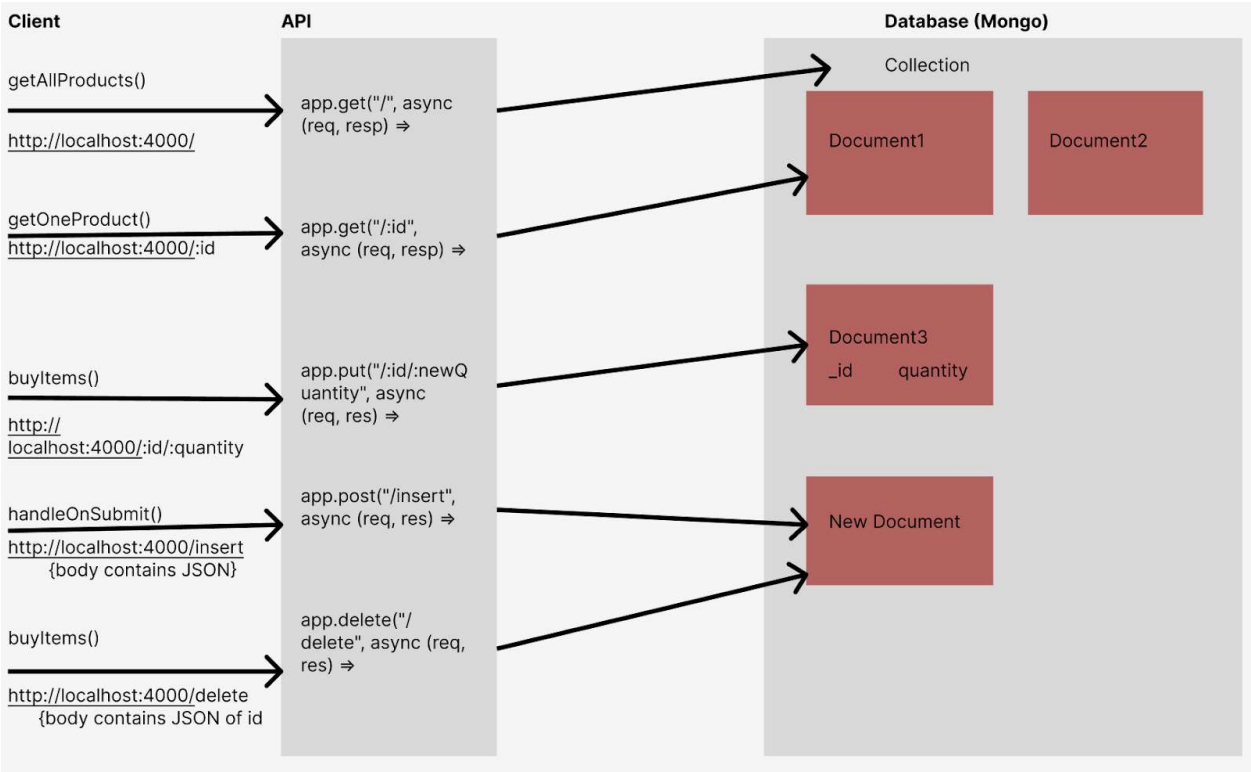
When the client first loads the website, it requests all the products from the server and reloads when it gets the response.

When the client clicks on the image or title of a product it requests the single product's information and reloads the detailed view when it gets the response.

When the client finishes a purchase, depending on the quantity of the product(s) left, it either updates the product's quantity or deletes the product from the database. If it requests the update to the product's quantity, the server will respond with the status of the update. If it requests the deletion of a product, the server will respond with the status of the deletion. After the confirmation view, the catalog is refreshed on the user's screen to show the updated amounts of the available products.

When the client finishes inputting the information required to add a new product to the catalog, the client machine generates a push request with all the information required to add the new object. Then the server adds it to the database and responds with the status of the addition. After clicking submit on the new product creation page, it brings the client to the catalog page and refreshes after the response from the server.

Descriptive text and graphics of the logical architecture



The above diagram shows how the logic flows from the client to the API to the Database. On the left are the functions that make the calls on the client, with the url they are calling to underneath the arrow. In the API box are the endpoints that handle the calls by the client. These calls handle the parameters and body of the calls and configure them for a correct query of the database. Then the API requests from the database.

The getAll endpoint requests all of the documents from the collection.

The getOne endpoint requests a certain document that matches the supplied _id.

The update endpoint updates a certain document that matches the supplied _id with the given updated quantity.

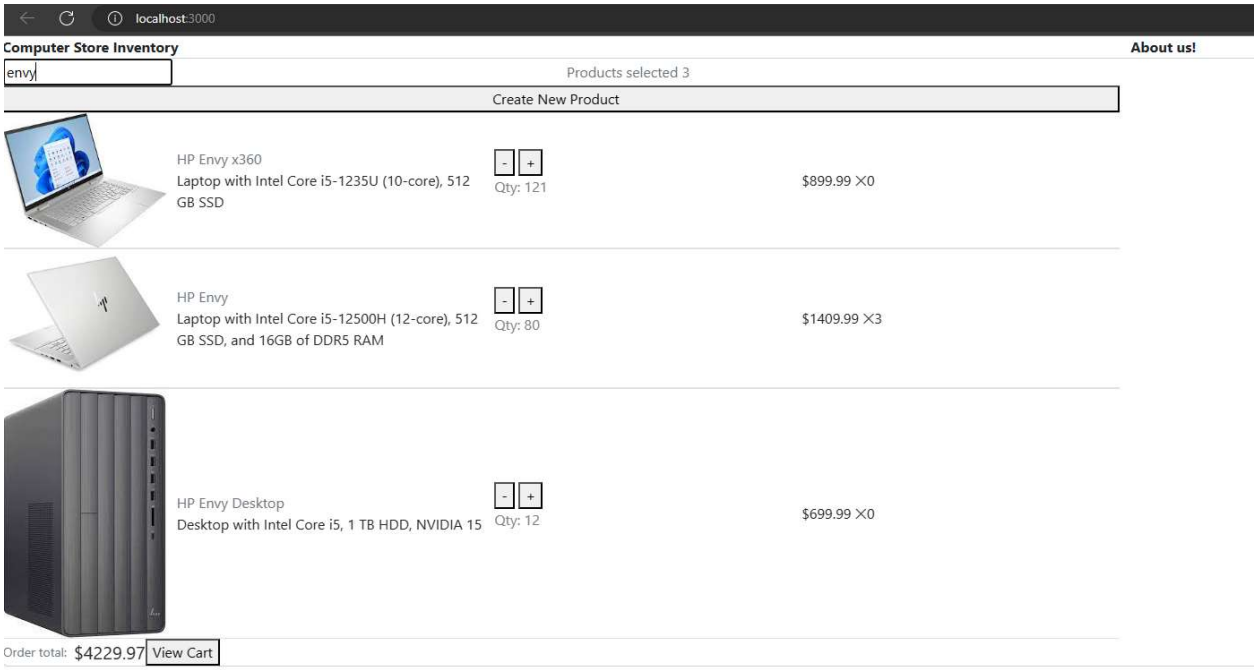
The insert endpoint inserts a new document into the database using the supplied JSON configuration in the body of the post method.

The delete endpoint deletes a document from the database using the supplied JSON configuration which holds the _id of the document to be deleted.

After finding and/or changing the document(s) requested, the database returns the document(s) that were found/changed/deleted to the API. The first two API methods, the getOne and getAll, return the document(s) to the client. The other API methods, the update, insert, and delete, simply return whether it was a success or failure since there isn't much need for the updated document. So all the arrows flip directions after the database finds or changes the document(s) requested to represent how the responses go.

View Explanations

Inventory View

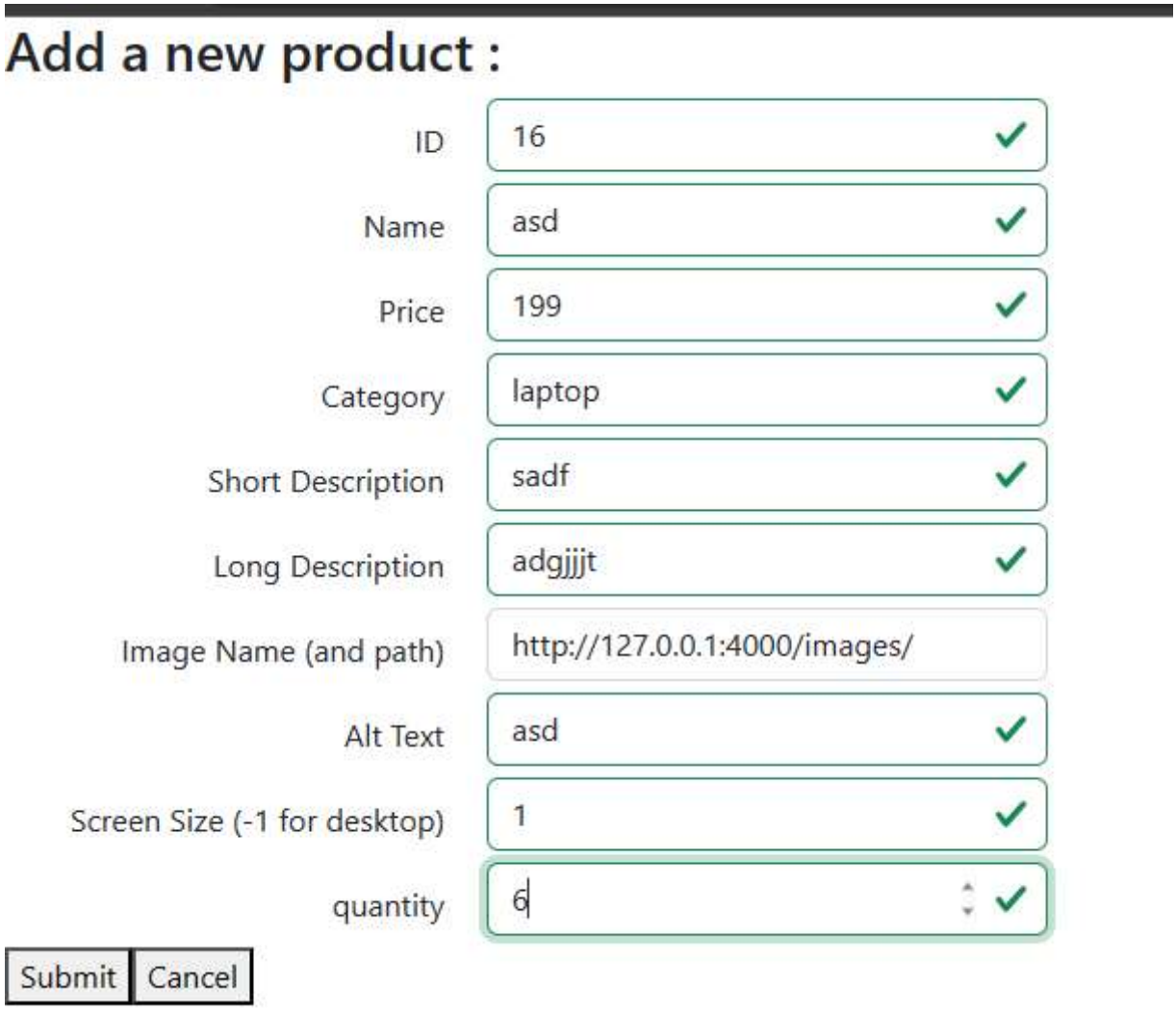


In this view the user can see what the computer store has in inventory. Each item displayed has '+' and '-' buttons that allow the user to add these items to their own cart. A counter is maintained to show how many items they have placed in their cart. If the user wishes to see more on a product they can either click on the image of the desired item or the name of the item. The user also has the ability to go to the Create Item view by clicking the 'Create New Products' button. In the top left there is a search bar that the user can use to narrow down their search to a specific name of computer. Clicking the 'about us' in the top right will send the user to our credits page. At the bottom of the display the user can see their current amount owed to the store if they wish to purchase what is in their cart. Clicking on the 'View Cart button will send the user to the cart display where they can fill a form to purchase their items.

Item View

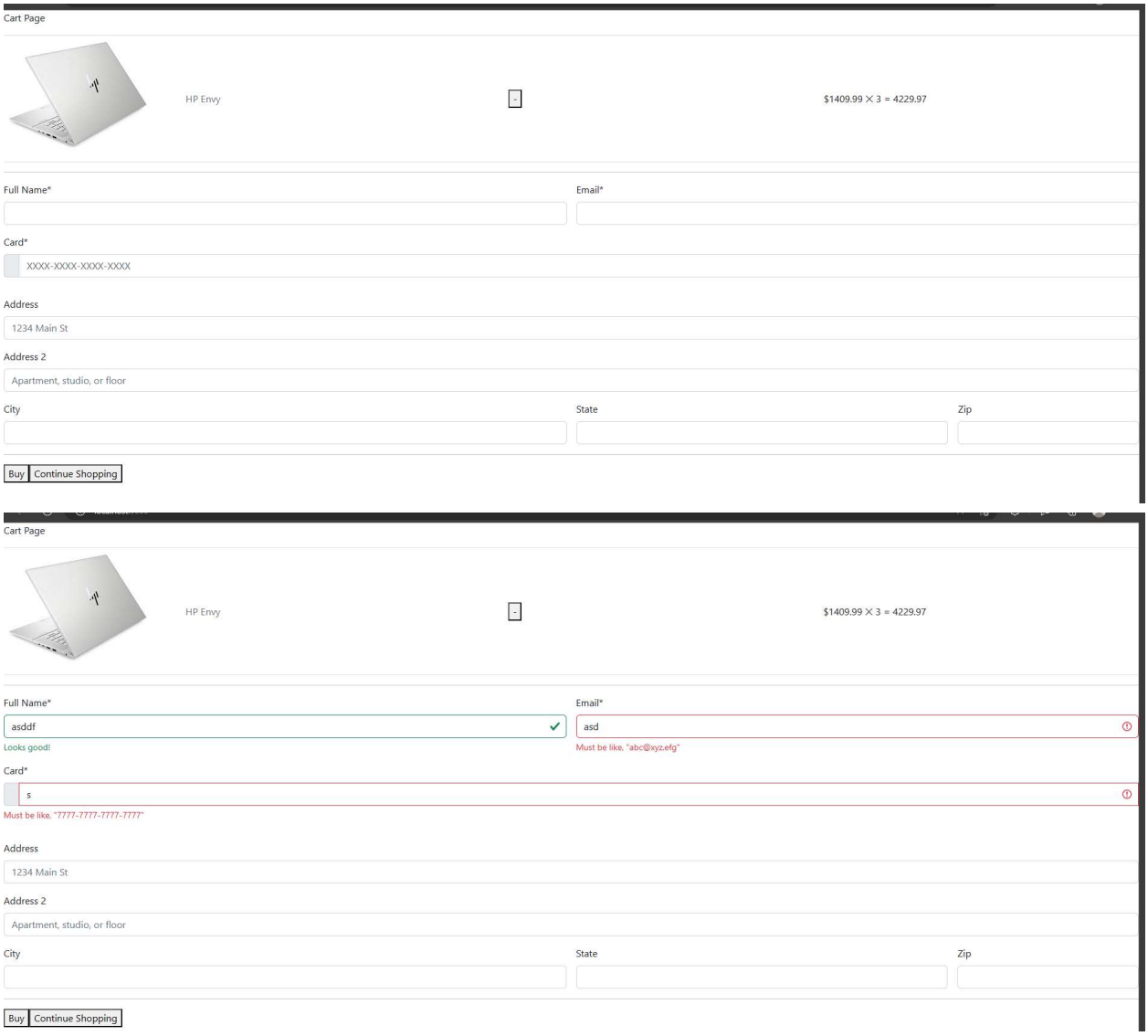
In this view the user is able to see more detail about a single item. They also gain the ability to add or remove this item from their cart with the '+' or '-' buttons. When the user clicks on the 'Keep Browsing' button, they will be sent back to the inventory view.

Create Item View



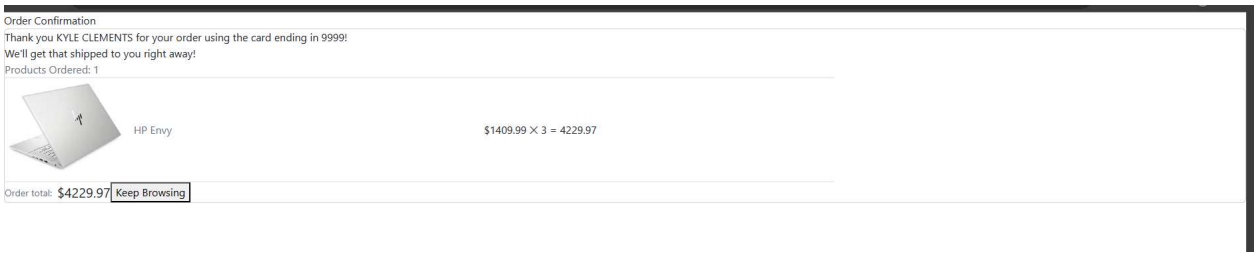
In this view the user can add a new product by filling out the fields provided for them. There are active checks for errors such as validating an id when entered. When the user is satisfied they can click the 'Submit' button. If the user wishes to abort, they can click on the 'Cancel' button to return to the Inventory view.

Cart View



In this view the user can see what items are currently in their cart. With the '-' button they do have the option to remove these items from their cart. To pay for these items the user must fill out the fields provided. There are checks on the 'Full Name', 'Email', and 'Card' fields to make sure the user puts in valid information. These checks are done when the user clicks on the 'Buy' button. If successful, they will be sent to the Confirmation view. The 'Continue Shopping' button sends the user to the Inventory view.

Confirmation View



In this view the user can see that their purchase has been successful. They are shown what they purchased and given the option to return to the inventory view with the 'Keep Browsing' button.

Manual of Installation

1) Install React
    a)
2) Install NodeJS
    a) Visit https://nodejs.org/en and install the latest version and follow the download instructions.
3) Install Express
    a) Run `npm install express` in your terminal.
4) Install CORS
    a)
5) Install Mongo and Mongo Compass
    a) Visit mongodb.com and download the latest version of the database and Mongo Compass and follow the download instructions.
6) Clone the GitHub repository from https://github.com/Nikben10/secoms319.
7) Start the backend database using `mongod`
    a) if on mac, use `mongod - -dbpath [Path to database]`
8) Change directory into the backend folder titled `46_backend`
9) Start the backend Node service using `nodemon index.js`
10) Change directory into the frontend folder titled `46_frontend`
11) Install the required packages using `npm install`
12) Start the frontend React using `npm start`
    Your browser should then have a page opened, showing the website.
13) If you are wanting to use our data for the catalog database :
    a) Open Mongo Compass and connect it to the default port, 27017.
    b) Create a new database called `reactdata`. If you already have one, then use that for the next steps.
    c) In that database, create a new collection titled `finalProject319`. If you already have one and want to replace its data, delete the collection and repeat step 8c and on.
    d) Click the `Add Data drop down and select `Import JSON or CSV file`
    e) Navigate to the `src` folder in the frontend and import `data.json`
    f) If nodemon is running, you'll need to refresh it by either stopping (Ctrl-c) it and starting (nodemon index.js) it again or by resaving a backend file, namely index.js in the backend folder.

Copy of Code

App.js (Frontend)

```
import './App.css';
import React, { useState, useEffect } from "react";

var userInfo = {
  fullName : '',
  cardNumber : '',
  city : '',
  zip : '',
  state : '',
  email : '',
  updateUser : function(fullname, cardNumber, email){
      this.fullName = fullname;
      this.cardNumber = cardNumber;
      this.email = email;
  }
}

let lowQuantity = 'red';
let normQuantity = 'grey';
let editedList = [];
let numAvailable = 10;

function App() {
  const [product, setProduct] = useState([]);
  const [cart, setCart] = useState([]);
  const [cartTotal, setCartTotal] = useState(0);
  const [page, setPage] = useState(0);            // 0 : Browse, 1 : Cart,
2 : Confirmation, 3: itemView , 5: credits
  const [oneProduct, setOneProduct] = useState([]);
  const [addNewProduct, setAddNewProduct] = useState({
    _id: 0,
    name: "",
    price: 0.0,
    category: "",
    shortdescription: "",
    longdescription: "",
```

```
    imagename: "http://127.0.0.1:4000/images/",
    alt: "",
    screensize: 0.0,
    quantity: 0,
  });

  const [searchString, setSearchString] = useState("");

  function getAllProducts() {
    fetch("http://localhost:4000/")
    .then((response) => response.json())
    .then((data) => {
        console.log("Show Catalog of Products :");
        console.log(data);
        if (product != data) {

            setProduct(data);
        }
    });
  }

  let itemsCopy = product;


  if (searchString.length > 0) {
    itemsCopy = product.filter((item) => {
        return
item.name.toUpperCase().includes(searchString.toUpperCase()) ||

item.shortDescription.toUpperCase().includes(searchString.toUpperCase());
    });
  }

  const listItems = itemsCopy.map((el) => (
    <div className="row border-top border-bottom" key={el._id}>
        <div className="row main align-items-center">
            <div className="col-2">
                <img className="img-fluid" src={el.imageName} alt={el.alt}
onClick={() => {setPage(3); getOneProduct(el._id);}}/>
            </div>
```

```
            <div className="col">
                <div className="row text-muted" onClick={() =>
{setPage(3); getOneProduct(el._id);}}>{el.name}</div>
                <div className="row">{el.shortDescription}</div>
            </div>
            <div className="col">
                <button type="button" variant="light" onClick={() =>
removeFromCart(el)} > - </button>{" "}
                <button type="button" variant="light" onClick={() =>
addToCart(el)}> + </button>
                <p style={{color: el.quantity > 5 ? normQuantity :
lowQuantity}}>
                    Qty: {el.quantity}
                </p>
            </div>
            <div className="col">
                ${el.price} <span
className="close">&#10005;</span>{howManyofThis(el)}
            </div>
        </div>
    </div>
    ));

  function howManyofThis(el) {
    for (let arr of cart) {
        if (arr._id == el._id) {
            return arr.cartQty;
        }
    }
    return 0;
  }

  useEffect(() => {
    total();
    }, [cart]);

  const total = () => {
    let totalVal = 0.00;
    for (let i = 0; i < cart.length; i++) {
        totalVal += Number(cart[i].price) * cart[i].cartQty;
```

```
  }
  setCartTotal(totalVal.toFixed(2));
};

const addToCart = (el) => {
  let hardCopy = [...cart];
  for (let arr of hardCopy) {
      if (arr._id == el._id) {
          // Found a match in array
          arr.cartQty++;
          setCart(hardCopy);
          return;
      }
  }
  el["cartQty"] = 1;
  hardCopy.push(el);
  setCart(hardCopy);
};

const removeFromCart = (el) => {
  let hardCopy = [...cart];
  for (let arr of hardCopy) {
      if (arr._id == el._id) {
          arr.cartQty--;
          if (arr.cartQty == 0) {
              hardCopy.splice(hardCopy.indexOf(arr), 1);
              break;
          }
      }
  }
  setCart(hardCopy);
};

function totalProducts() {
  let total = 0;
  for (let i of cart) {
      total += i.cartQty;
  }
  return total;
}
```

```
  const showOneItem = oneProduct.map((el) => (
    <div className="cover-container d-flex w-100 h-100 p-3 mx-auto
flex-column">
        <header className="mb-auto">
            <img className="img-fluid" src={el.imageName}></img>
        </header>
        <main className="px-3">
            <h1><b>{el.name}</b></h1>
                    <p>{el.longDescription}</p>
                    <p color="green">{el.price}</p>
        </main>

        <footer>
            <div><p>How many should we add to cart?</p></div>
            <div className="row g-0 text-center">
                <div className="col-sm-4 col-md-4">
                    <button type="button" variant="light" onClick={() =>
removeFromCart(el)}> - </button>
                </div>
                <div className="col-sm-4 col-md-4">
                    {howManyofThis(el)}
                </div>
                <div className="col-4 col-md-4">
                    <button type="button" variant="light" onClick={() =>
addToCart(el)}> + </button>
                </div>
            </div>
        </footer>
    </div>
  ));

  function getOneProduct(id) {
    console.log(id);
    if (id >= 1 && id <= 20) {
      fetch("http://localhost:4000/" + id)
      .then((response) => response.json())
      .then((data) => {
        console.log("Show one product :", id);
        console.log(data);
```

```
        const dataArr = [];
        dataArr.push(data);
        setOneProduct(dataArr);
      });
    } else {
      console.log("Wrong number of Product id.");
      console.log('We sent %d', id);
      setOneProduct([]);
    }
  }

  const cartItems = cart.map((el) => (
    <div className="row border-top border-bottom" key={el._id}>
        <div className="row main align-items-center">
            <div className="col-2">
                <img className="img-fluid" src={el.imageName} />
            </div>
            <div className="col">
                <div className="row text-muted">{el.name}</div>
            </div>
            <div className="col">
                <button type="button" variant="light" onClick={() =>
removeFromCart(el)} > - </button>{" "}
            </div>
            <div className="col">
                ${el.price} <span className="close">&#10005;</span>
{el.cartQty} = {(el.price * el.cartQty).toFixed(2)}
            </div>
        </div>
    </div>
));

const checkoutedItems = cart.map((el) => (
  <div className="row border-top border-bottom" key={el._id}>
      <div className="row main align-items-center">
          <div className="col-2">
              <img className="img-fluid" src={el.imageName} />
          </div>
          <div className="col">
              <div className="row text-muted">{el.name}</div>
```

```
          </div>
          <div className="col">
              ${el.price} <span className="close">&#10005;</span>
{el.cartQty} = {(el.price * el.cartQty).toFixed(2)}
          </div>
      </div>
  </div>
));

function buyItems(cart) {
    cart.forEach(el => {
        let newQuantity = el.quantity - el.cartQty;
        if (newQuantity > 0) {
            console.log("Updating id : " + el._id + " to have quantity : "
+ newQuantity);
            fetch("http://localhost:4000/" + el._id + "/" + newQuantity, {
                method: "PUT"
            })
            .then((response) => response.json())
            .then((data) => {
                console.log("Updated quantity :");
                console.log(data);
            });
        } else {
            // Delete the product
            fetch("http://localhost:4000/delete/", {
                method: "DELETE",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ _id: el._id }),
            })
            .then((response) => response.json())
            .then((data) => {
                console.log("Deleted prooduct : " + el._id);
                console.log(data);
            });
        }
    });
}

const alertPlaceholder = document.getElementById('liveAlertPlaceholder');
```

```
const form = document.getElementById('checkout-form');
const summaryCard = document.querySelector('.card');
const summaryList = document.querySelector('.card > ul');


let validate = function(){
  let val = true;
  let email = document.getElementById('inputEmail')
  let name = document.getElementById('inputName')
  let card = document.getElementById('inputCard')


  if (name.value.length == 0)
  {
  name.setAttribute("class","form-control is-invalid")
  val = false;
  }
  else{
  name.setAttribute("class", "form-control is-valid");
  userInfo.name = name.value;
  }
  if (!isNaN(card.value) || card.value.length < 16)
  {
  card.setAttribute("class","form-control is-invalid");
  val = false;
  }
  else{
  card.setAttribute("class", "form-control is-valid");
  userInfo.card = card.value;
  }
  if (!email.value.includes("@")){
    email.setAttribute("class", "form-control is-invalid");
    val = false;
  }
  else{
    email.setAttribute("class", "form-control is-valid");
    userInfo.email = email.value;
  }
  if (val){
      userInfo.updateUser(name.value, card.value, email.value);
```

```
  }
  if(val){
        buyItems(cart);
      setPage(2);
  }
  else{
      setPage(1);
  }
}

const alert = (message, type) => {
  const wrapper = document.createElement('div')
  wrapper.innerHTML = [
      `<div class="alert alert-${type} alert-dismissible" role="alert">`,
      `  <div>${message}</div>`,
      ' <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>',
      '</div>'
      ].join('')

      alertPlaceholder.append(wrapper);
}

function handleChange(evt) {
    const value = evt.target.value;
    document.getElementById("correctIt").hidden = true;
    document.getElementById("setIt").hidden = true;
    if (evt.target.name === "_id") {
        setAddNewProduct({ ...addNewProduct, _id: value});
        if (!editedList.includes("_id")) {
            editedList.push("_id");
        }
        if (value < 1) {
            document.getElementById("_id").setAttribute("class",
"form-control is-invalid");
            return;
        }
        for (let prod of product) {
            if (prod._id == value) {
```

```
                document.getElementById("_id").setAttribute("class",
"form-control is-invalid");
                return;
            }
        }
        document.getElementById("_id").setAttribute("class", "form-control
is-valid");
        return;
    } else if (evt.target.name === "name") {
        setAddNewProduct({ ...addNewProduct, name: value});
        if (!editedList.includes("name")) {
            editedList.push("name");
        }
        if (value.length < 1) {
            document.getElementById("name").setAttribute("class",
"form-control is-invalid");
            return;
        }
        document.getElementById("name").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "price") {
        setAddNewProduct({ ...addNewProduct, price: value});
        if (!editedList.includes("price")) {
            editedList.push("price");
        }
        if (value < 0 || value.length == 0) {
            document.getElementById("price").setAttribute("class",
"form-control is-invalid");
            return;
        }
        document.getElementById("price").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "category") {
        setAddNewProduct({ ...addNewProduct, category: value});
        if (!editedList.includes("category")) {
            editedList.push("category");
        }
        if (value.length < 1) {
```

```
            document.getElementById("category").setAttribute("class",
"form-control is-invalid");
            return;
        }
        document.getElementById("category").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "shortDescription") {
        setAddNewProduct({ ...addNewProduct, shortdescription: value});
        if (!editedList.includes("shortDesc")) {
            editedList.push("shortDesc");
        }
        if (value.length < 1) {

document.getElementById("shortDescription").setAttribute("class",
"form-control is-invalid");
            return;
        }
        document.getElementById("shortDescription").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "imageName") {
        setAddNewProduct({ ...addNewProduct, imagename: value});
        if (!editedList.includes("imageName")) {
            editedList.push("imageName");
        }
        document.getElementById("imageName").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "longDescription") {
        setAddNewProduct({ ...addNewProduct, longdescription: value});
        if (!editedList.includes("longDesc")) {
            editedList.push("longDesc");
        }
        if (value.length < 1) {

document.getElementById("longDescription").setAttribute("class",
"form-control is-invalid");
            return;
        }
```

```
        document.getElementById("longDescription").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "alt") {
        setAddNewProduct({ ...addNewProduct, alt: value});
        if (!editedList.includes("alt")) {
            editedList.push("alt");
        }
        if (value.length < 1) {
            document.getElementById("alt").setAttribute("class",
"form-control is-invalid");
            return;
        }
        document.getElementById("alt").setAttribute("class", "form-control
is-valid");
        return;
    } else if (evt.target.name === "screenSize") {
        setAddNewProduct({ ...addNewProduct, screensize: value});
        if (!editedList.includes("screenSize")) {
            editedList.push("screenSize");
        }
        if (value < -1 || value.length == 0) {
            document.getElementById("screenSize").setAttribute("class",
"form-control is-invalid");
            return;
        }
        document.getElementById("screenSize").setAttribute("class",
"form-control is-valid");
        return;
    } else if (evt.target.name === "quantity") {
        setAddNewProduct({ ...addNewProduct, quantity: value});
        if (!editedList.includes("quantity")) {
            editedList.push("quantity");
        }
        if (value < 1 || value.length == 0) {
            document.getElementById("quantity").setAttribute("class",
"form-control is-invalid");
            return;
        }
```

```
        document.getElementById("quantity").setAttribute("class",
"form-control is-valid");
        return;
    }
}

function handleOnSubmit(e) {
    e.preventDefault();
    console.log(e.target.value);
    if (editedList.length != numAvailable) {
        console.log("Enter more");
        document.getElementById("setIt").hidden = false;
        return;
    }
    let list = ["_id", "name", "price", "category", "shortDescription",
"longDescription", "imageName", "alt", "screenSize", "quantity"];
    let correct = true;
    list.forEach(id => {
        let element = document.getElementById(id);
        if (!element.classList.contains("is-valid")) {
            console.log("correct more");
            document.getElementById("correctIt").hidden = false;
            correct = false;
            return;
        }
    });
    if (correct) {
    fetch("http://localhost:4000/insert", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(addNewProduct),
    })
    .then((response) => response.json())
    .then((data) => {
        console.log("Post a new product completed");
        console.log(data);
        if (data) {
            getAllProducts();
            editedList = [];
            setAddNewProduct({
```

```
                _id: 0,
                name: "",
                price: 0.0,
                category: "",
                shortdescription: "",
                longdescription: "",
                imagename: "http://127.0.0.1:4000/images/",
                alt: "",
                screensize: 0.0,
                quantity: 0,
              });
            document.getElementById("createForm").reset();
            setPage(0);
        }
    });
}
}


if (page == 0) {
    if(product.length < 1){
        getAllProducts();
    }
  return (
  <div>
    <div className="row">
        <div className="col"><b>Computer Store Inventory</b></div>
        <div className="col"></div>
        <div className="col">
            <b onClick={() => setPage(5)}>About us!</b>
        </div>
    </div>
      <div className="card">
          <div className="row">
          {/* HERE, IT IS THE SHOPING CART */}
              <div className="col-md-8 cart">
                  <div className="title">
                      <div className="row">
                          <div className="col">
                              <input onChange={(e) => {
```

```
                                setSearchString(e.target.value);
                                }} placeholder="Search">
                            </input>
                        </div>
                        <div className="col align-self-center text-right
text-muted">
                            Products selected {totalProducts()}
                        </div>
                          <button type="button" variant="light"
onClick={() => setPage(4)}>Create New Product</button>
                    </div>
                </div>
                <div>{listItems}</div>
            </div>
            <div className ="float-end">
                <p className ="mb-0 me-5 d-flex align-items-center">
                    <span className ="small text-muted me-2">Order
total:</span>
                    <span className ="lead
fw-normal">${cartTotal}</span>
                    <button type="button" variant="light" onClick={() =>
setPage(1)}> View Cart </button>
                </p>
            </div>
        </div>
    </div>
  </div>
  );
}else if (page == 1) {
  return (
    <div>
        <p>
            Cart Page
        </p>
        <div>{cartItems}</div>
        <hr></hr>
        <div id="liveAlertPlaceholder"></div>
        <form className="row g-3" id="checkout-form">
            <div className="col-md-6">
```

```
                    <label htmlFor="inputName" className="form-label">Full
Name*</label>
                    <input type="text" className="form-control"
id="inputName"></input>
                    <div className="valid-feedback">
                        Looks good!
                    </div>
                    <div className="invalid-feedback">
                        Must be like, "John Doe"
                    </div>
                </div>

                <div className="col-md-6">
                    <label htmlFor="inputEmail"
className="form-label">Email*</label>
                    <input type="email" className="form-control"
id="inputEmail"></input>
                    <div className="valid-feedback">
                        Looks good!
                    </div>
                    <div className="invalid-feedback">
                        Must be like, "abc@xyz.efg"
                    </div>
                </div>

                <div className="col-12">
                    <label htmlFor="inputCard"
className="form-label">Card*</label>
                    <div className="input-group mb-3">
                        <span className="input-group-text"
id="basic-addon1"><i className="bi-credit-card-fill"></i></span>
                        <input type="text" id="inputCard"
className="form-control" placeholder="XXXX-XXXX-XXXX-XXXX"
aria-label="Username" aria-describedby="basic-addon1"></input>
                        <div className="invalid-feedback">
                        Must be like, "7777-7777-7777-7777"
                        </div>
                    </div>
                </div>
```

```
              <div className="col-12">
                  <label htmlFor="inputAddress"
className="form-label">Address</label>
                  <input type="text" className="form-control"
id="inputAddress" placeholder="1234 Main St"></input>
              </div>
              <div className="col-12">
                  <label htmlFor="inputAddress2"
className="form-label">Address 2</label>
                  <input type="text" className="form-control"
id="inputAddress2" placeholder="Apartment, studio, or floor"></input>
              </div>
              <div className="col-md-6">
                  <label htmlFor="inputCity"
className="form-label">City</label>
                  <input type="text" className="form-control"
id="inputCity"></input>
              </div>
              <div className="col-md-4">
                  <label htmlFor="inputState"
className="form-label">State</label>
                  <input type="text" className="form-control"
id="inputCity"></input>
              </div>
              <div className="col-md-2">
                  <label htmlFor="inputZip"
className="form-label">Zip</label>
                  <input type="text" className="form-control"
id="inputZip"></input>
              </div>
          </form>
          <hr></hr>
          <button type="button" variant="light" onClick={() =>
validate()}> Buy </button>
          <button type="button" variant="light" onClick={() =>
setPage(0)}> Continue Shopping</button>
      </div>
  );
}else if (page == 2){
  return (
```

```
  <div>
      Order Confirmation
      <div className="card">
          <div className="row">
              <div className="col-md-8 cart">
                  <div className="title">
                      {/* <div className="row"> */}
                          <div className="col">
                              Thank you {userInfo.fullName} for your order
using the card ending in {userInfo.cardNumber.slice(-4)}!<br />
                              We'll get that shipped to {userInfo.city !=
'' ? userInfo.city : "you"} right away!
                          </div>
                          <div className="col align-self-center text-right
text-muted">
                              Products Ordered: {cart.length}
                          </div>
                      {/* </div> */}
                  </div>
                  <div>{checkoutedItems}</div>
              </div>
              <div className ="float-end">
                  <p className ="mb-0 me-5 d-flex align-items-center">
                      <span className ="small text-muted me-2">Order
total:</span>
                      <span className ="lead
fw-normal">${cartTotal}</span>
                      <button type="button" variant="light" onClick={() =>
{setCart([]); getAllProducts(); setPage(0)}}> Keep Browsing </button>
                  </p>
              </div>
          </div>
      </div>
  </div>
  );
} else if (page == 3) {
    return(
        <div className="d-flex h-200 text-center text-bg-dark">
            <div>{showOneItem}</div>
```

```
            <button type="button" variant="light" className="d-flex h-100
text-right" onClick={() => {setPage(0)}}>Keep Browsing</button>
        </div>
    )
} else if (page == 4) {
    return (
        <div>
          <h3>Add a new product :</h3>
          <form id="createForm">
            <div className='form-group row'>
                <label className='col-sm-2 col-form-label
createLabel'>ID</label>
                <div className='col-sm-2'>
                    <input type="number" placeholder="id"
className='form-control' id="_id" name="_id" value={addNewProduct._id}
onChange={handleChange} />
                </div>
            </div>
            <div className='form-group row'>
                <label className='col-sm-2 col-form-label
createLabel'>Name</label>
                <div className='col-sm-2'>
                    <input type="text" placeholder="name"
className='form-control' id="name" name="name" value={addNewProduct.name}
onChange={handleChange} />
                </div>
            </div>
            <div className='form-group row'>
                <label className='col-sm-2 col-form-label
createLabel'>Price</label>
                <div className='col-sm-2'>
                    <input type="number" placeholder="price"
className='form-control' id="price" name="price"
value={addNewProduct.price} onChange={handleChange} />
                </div>
            </div>
            <div className='form-group row'>
                <label className='col-sm-2 col-form-label
createLabel'>Category</label>
                <div className='col-sm-2'>
```

```
                    <input type="text" placeholder="category"
classNameName='form-control' id="category" name="category"
value={addNewProduct.category} onChange={handleChange} />
                </div>
            </div>
            <div classNameName='form-group row'>
                <label classNameName='col-sm-2 col-form-label
createLabel'>Short Description</label>
                <div classNameName='col-sm-2'>
                    <input type="text" placeholder="Short Description"
classNameName='form-control' id="shortDescription" name="shortDescription"
value={addNewProduct.shortdescription} onChange={handleChange} />
                </div>
            </div>
            <div classNameName='form-group row'>
                <label classNameName='col-sm-2 col-form-label
createLabel'>Long Description</label>
                <div classNameName='col-sm-2'>
                    <input type="text" placeholder="Long description"
classNameName='form-control' id="longDescription" name="longDescription"
value={addNewProduct.longdescription} onChange={handleChange} />
                </div>
            </div>
            <div classNameName='form-group row'>
                <label classNameName='col-sm-2 col-form-label
createLabel'>Image Name (and path)</label>
                <div classNameName='col-sm-2'>
                    <input type="text" placeholder="image name"
classNameName='form-control' id="imageName" name="imageName"
value={addNewProduct.imagename} onChange={handleChange} />
                </div>
            </div>
            <div classNameName='form-group row'>
                <label classNameName='col-sm-2 col-form-label createLabel'>Alt
Text</label>
                <div classNameName='col-sm-2'>
                    <input type="text" placeholder="alt text"
classNameName='form-control' id="alt" name="alt" value={addNewProduct.alt}
onChange={handleChange} />
                </div>
```

```
            </div>
            <div className='form-group row'>
                <label className='col-sm-2 col-form-label
createLabel'>Screen Size (-1 for desktop)</label>
                <div className='col-sm-2'>
                    <input type="number" placeholder="screen size"
className='form-control' id="screenSize" name="screenSize"
value={addNewProduct.screensize} onChange={handleChange} />
                </div>
            </div>
            <div className='form-group row'>
                <label className='col-sm-2 col-form-label
createLabel'>quantity</label>
                <div className='col-sm-2'>
                    <input type="number" placeholder="quantity"
className='form-control' id="quantity" name="quantity"
value={addNewProduct.quantity} onChange={handleChange} />
                </div>
            </div>
            <div>
              <button type="submit" onClick={handleOnSubmit}>
                  Submit
              </button>
              <button onClick={() => {setPage(0); editedList = []}}>
                  Cancel
              </button>
              <p id="correctIt" style={{color: 'red'}} hidden>
                Please correct the required inputs.
              </p>
              <p id="setIt" style={{color: 'red'}} hidden>
                Please set all the inputs.
              </p>
            </div>
          </form>
        </div>
    )
} else{
    return(
        <div className="container text-center">
            <div className="row">
```

```
                <div className="col"></div>
                <div className="col">
                    <h2>SE/ComS319 Construction of User Interfaces, Spring
2023</h2>
                </div>
                <div className="col">
                    <button type="button" variant="light" onClick={() =>
{setPage(0)}}>Back to Shopping!</button>
                </div>
            </div>
            <div className="row">
                <p>Date: 5/4/2023</p>
            </div>
            <hr></hr>
            <div className="row"><h3>Authors:</h3></div>
            <div className="row">
                <div className="col">
                    <h2><b>Benjamin Niklasen</b></h2>
                </div>
                <div className="col">
                    <h2>bjn1@iastate.edu</h2>
                </div>
            </div>
            <div className="row">
                <div className="col">
                    <h2><b>Kyle Clements</b></h2>
                </div>
                <div className="col">
                    <h2>kyle0324@iastate.edu</h2>
                </div>
            </div>
            <div className="row"> <hr></hr></div>
            <div className="row">
                <h3>Proffessor: </h3>
            </div>
            <div className="row">
                <div className="col">
                    <h2>Dr. Abraham N. Aldaco Gastelum</h2>
                </div>
                <div className="col">
```

```
                    <h2>aaldaco@iastate.edu</h2>
                </div>
            </div>
            <hr></hr>
            <div className="row">
                <p>Thank you for viewing our product!</p>
            </div>
        </div>
    )
}
}

export default App;
```

Index.js (frontend)

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import "bootstrap/dist/css/bootstrap.css";
// import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Index.js (backend)

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const app = express();
const Product = require("./dataSchema.js");

app.use(express.json());
app.use(cors());

app.use(express.static("public"));
app.use("/images", express.static("images"));

mongoose.connect("mongodb://127.0.0.1:27017/reactdata",
    {
        dbName: "reactdata",
        useNewUrlParser: true,
        useUnifiedTopology: true,
    }
);

app.get("/", async (req, resp) => {
    const query = {};
    const allProducts = await Product.find(query);
    console.log(allProducts);
    resp.send(allProducts);
});

app.get("/:id", async (req, resp) => {
    const id = req.params.id;
    const query = { _id: id };
    const oneProduct = await Product.findOne(query);
    console.log(oneProduct);
    resp.send(oneProduct);
});
```

```
app.put("/:id/:newQuantity", async (req, res) => {
    try {
        // await formData.save();
        const filter = { _id: req.params.id };
        const update = { quantity: Number(req.params.newQuantity) };
        console.log("New quantity: update");
        await Product.findOneAndUpdate(filter, update);
        const messageResponse = { message: `Product ${req.params.id}
updated correctly`};
        res.send(JSON.stringify(messageResponse));
    } catch (err) {
        console.log("Error while updating a product:" + err);
    }
});

app.post("/insert", async (req, res) => {
    console.log(req.body);
    const p_id = req.body._id;
    const pname = req.body.name;
    const pprice = req.body.price;
    const pquantity = req.body.quantity;
    const pshortdescription = req.body.shortdescription;
    const plongdescription = req.body.longdescription;
    const pcategory = req.body.category;
    const pimagename = req.body.imagename;
    const palt = req.body.alt;
    const pscreensize = req.body.screensize;
    const formData = new Product({
        _id: p_id,
        name: pname,
        price: pprice,
        shortDescription: pshortdescription,
        longDescription: plongdescription,
        category: pcategory,
        imageName: pimagename,
        quantity: pquantity,
        alt: palt,
        screenSize: pscreensize
    });
    try {
```

```
        // await formData.save();
        await Product.create(formData);
        const messageResponse = { message: `Product ${p_id} added
correctly` };
        res.send(JSON.stringify(messageResponse));
    } catch (err) {
        console.log("Error while adding a new product:" + err);
    }
});

app.delete("/delete", async (req, res) => {
    console.log("Delete :", req.body);
    try {
        const query = { _id: req.body._id };
        await Product.deleteOne(query);
        const messageResponse = {
            message: `Product ${req.body._id} deleted correctly`,
        };
        res.send(JSON.stringify(messageResponse));
    } catch (err) {
        console.log("Error while deleting :" + p_id + " " + err);
    }
});

const port = process.env.PORT || 4000;
const host = "localhost";
app.listen(port, () => {
    console.log(`App listening at http://%s:%s`, host, port);
});
```

dataSchema.js (backend)

```
const mongoose = require('mongoose');

const ReactFormDataSchema = new mongoose.Schema({
        _id: {type: Number},
        name: {type: String},
        price: {type: Number},
        category: {type: String},
        shortDescription: {type: String},
        longDescription: {type: String},
        imageName: {type: String},
        alt: {type: String},
        screenSize: {type: Number},
        quantity: {type: Number}
    },
    {
    collection: "finalProject319"
    }
);

const Product = mongoose.model('Product', ReactFormDataSchema);

module.exports = Product;
```

finalProject319.json


```
[{
  "_id": 1,
  "name": "HP Spectre x360",
  "price": 1249.99,
  "category": "Laptop",
  "shortDescription": "2-in-1 laptop with Intel Core i5-1235U (10-core), 512 GB SSD",
  "longDescription": "This 2-in-1 laptop has it all with an i5 Intel Core processor with up to 4.4 GHz and 10 cores, 512GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_spectre_x360_13.5.jpeg",
  "screenSize": 13.5,
  "alt": "HP Spectre x360",
  "quantity": 55
},{
  "_id": 2,
  "name": "HP Envy x360",
  "price": 899.99,
  "category": "Laptop",
  "shortDescription": "Laptop with Intel Core i5-1235U (10-core), 512 GB SSD",
  "longDescription": "This laptop has most of it with an i5 Intel Core processor with up to 4.4 GHz and 10 cores, 512GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_envy_x360_13.3.jpeg",
  "screenSize": 13.3,
  "alt": "HP Envy x360",
  "quantity": 123
},{
  "_id": 3,
  "name": "HP Envy",
  "price": 1409.99,
  "category": "Laptop",
  "shortDescription": "Laptop with Intel Core i5-12500H (12-core), 512 GB SSD, and 16GB of DDR5 RAM",
  "longDescription": "This laptop has it all with an i5 Intel Core processor with up to 4.5 GHz and 12 cores, 512GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_envy_16.jpeg",
```

```
  "screenSize": 16,
  "alt": "HP Envy",
  "quantity": 83
},{
  "_id": 4,
  "name": "HP Pavilion",
  "price": 449.99,
  "category": "Laptop",
  "shortDescription": "Laptop with Intel Core i3-1215U (6-core), 256 GB
SSD",
  "longDescription": "This laptop has some of it with an i3 Intel Core
processor with up to 4.4 GHz and 6 cores, 256GB SSD storage, and Windows
11.",
  "imageName": "http://127.0.0.1:4000/images/hp_pavilion_14.jpeg",
  "screenSize": 14,
  "alt": "HP Pavilion",
  "quantity": 75
},{
  "_id": 5,
  "name": "HP 14\" laptop",
  "price": "549.99",
  "category": "Laptop",
  "shortDescription": "Laptop with Intel Core i5-1235U (10-core), 256 GB
SSD",
  "longDescription": "This laptop has some of it with an i5 Intel Core
processor with up to 4.4 GHz and 10 cores, 256GB SSD storage, and Windows
11.",
  "imageName": "http://127.0.0.1:4000/images/hp_14_laptop.jpeg",
  "screenSize": "14",
  "alt": "HP 14\" laptop",
  "quantity": 52
},{
  "_id": 6,
  "name": "HP Omen",
  "price": 34.5,
  "category": "Laptop",
  "shortDescription": "Laptop with Intel Core i3-12500H (12-core), 512 GB
SSD",
```

```
  "longDescription": "This laptop has some of it with an i3 Intel Core processor with up to 4.5 GHz and 12 cores, 512GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_omen_16.jpeg",
  "screenSize": 16,
  "alt": "HP Omen",
  "quantity": 18
},{
  "_id": 7,
  "name": "HP Chromebase All-in-One",
  "price": 1569.99,
  "category": "All-in-One",
  "shortDescription": "All-in-One desktop with Intel Core i5-12500H (12-core), 1TB SSD",
  "longDescription": "This All-in-One desktop has some of it with an i5 Intel Core processor with up to 4.5 GHz and 12 cores, 1TB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_chromebase_22.jpeg",
  "screenSize": 22,
  "alt": "HP Chromebase All-in-One",
  "quantity": 27
},{
  "_id": 8,
  "name": "HP Envy Desktop",
  "price": 699.99,
  "category": "Desktop",
  "shortDescription": "Desktop with Intel Core i5, 1 TB HDD, NVIDIA 15",
  "longDescription": "This desktop has most of it with an i5 Intel Core processor with an NVIDIA GeForce 1660 Super, 1TB HDD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_envy_desktop.jpeg",
  "screenSize": -1,
  "alt": "HP Envy Desktop",
  "quantity": 12
},{
  "_id": 9,
  "name": "HP Pavilion Desktop",
  "price": 829.99,
  "category": "Desktop",
  "shortDescription": "Desktop with Intel Core i7, 2TB HDD, 512GB SSD",
```

```
  "longDescription": "This laptop has it all with an i7 Intel Core processor, 2TB HDD and 512GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_pavilion_desktop.jpeg",
  "screenSize": -1,
  "alt": "HP Pavilion Desktop",
  "quantity": 909
},{
  "_id": 10,
  "name": "HP Omen 25L 2020",
  "price": 1379.99,
  "category": "Desktop",
  "shortDescription": "Desktop with Intel Core i5-12400 (6-core), 256 GB SSD",
  "longDescription": "This desktop has some of it with an i5 Intel Core processor with up to 4.4 GHz and 6 cores, 256GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_omen_desktop.jpeg",
  "screenSize": -1,
  "alt": "HP Omen 25L 2020",
  "quantity": 41
},{
  "_id": 11,
  "name": "HP Victus 15",
  "price": 829.99,
  "category": "Desktop",
  "shortDescription": "Desktop with Intel Core i5-12400 (12-core), 256 GB SSD",
  "longDescription": "This desktop has some of it with an i5 Intel Core processor with up to 4.4 GHz and 12 cores, 256GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_victus_desktop.jpeg",
  "screenSize": -1,
  "alt": "HP Victus 15",
  "quantity": 5
},{
  "_id": 12,
  "name": "HP 24 All-in-One",
  "price": 799.99,
  "category": "All-in-One",
```

```
  "shortDescription": "All-in-One with Intel Core i3-13100T (4-core), 256 GB SSD",
  "longDescription": "This All-in-One desktop has it all with an i3 Intel Core processor with up to 4.2 GHz and 4 cores, 256GB SSD storage, and Windows 11.",
  "imageName": "http://127.0.0.1:4000/images/hp_24_AiO.jpeg",
  "screenSize": "24",
  "alt": "HP 24 All-in-One",
  "quantity": 74
}]
```