Computer Science 319 :

Construction of User Interfaces

5/1/2023

Kaden Wingert: kadenwin@iastate.edu

Bryce Maloy: bsmaloy@iastate.edu

Professor: Abraham Aldaco

Final Project Documentation

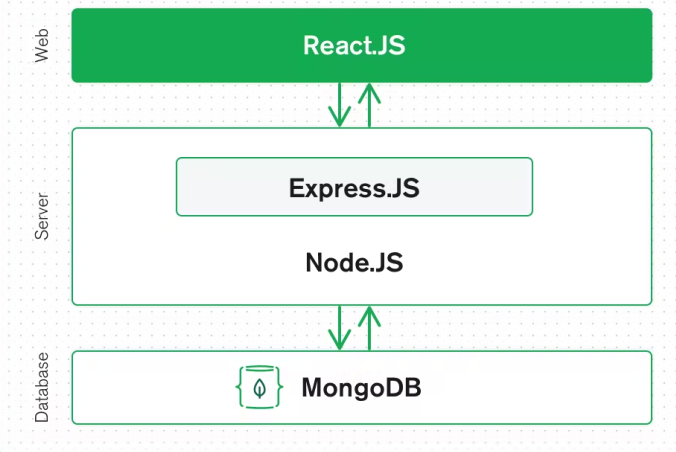
Index

Project Description

Our final project for this class includes a single webpage containing a catalog of products developed for a small business: Nordland Forge. General features of this webpage include the ability to show all of the products, as well as adding removing, and updating any product. This information is stored in a MongoDB database, and any changes made to products are done there. Along with using Mongo, we used Express, Node, and Javascript in the backend. The front end uses software such as React and CSS. In the front end, we implemented a footer that allows one to view an About page, which explains the origin of this business.
Additionally, the footer contains a credits page that documents the website's creators and the class it was created for.

Diagram of Project

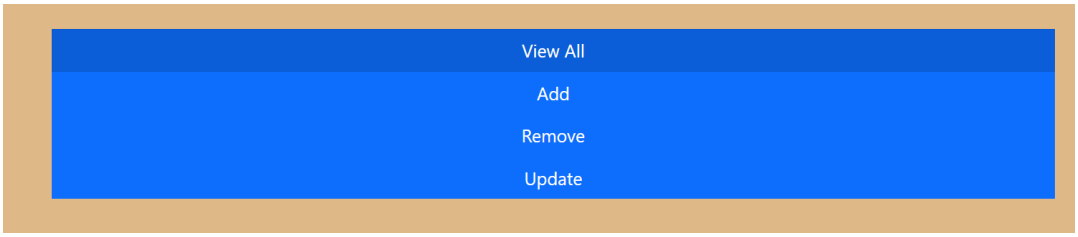
The overall flow of this project includes using react in the front end, and when you attempt to access the database through POST/PUT, GET, or DELETE, you attempt to fetch data from the server, which then accesses the Mongo database. Then the relevant database's information is sent back to the server and back to the frontend.



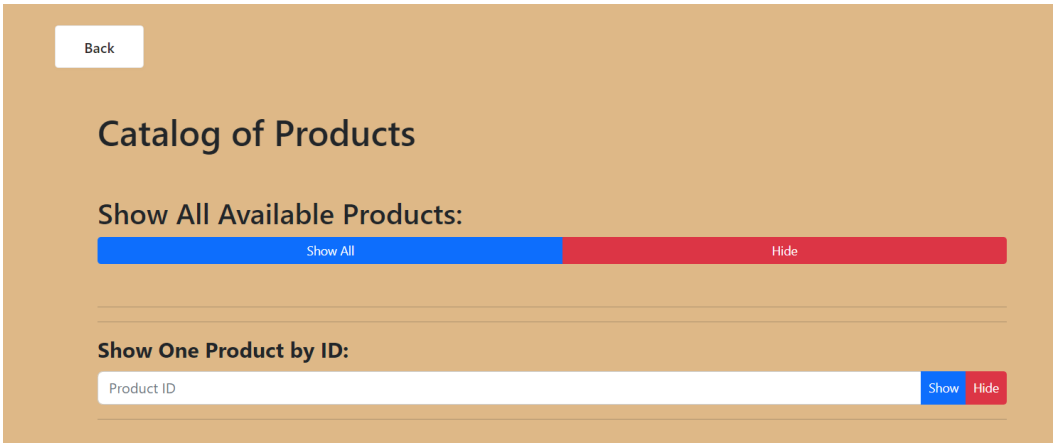
In the frontend, the navigation from each view is done through hooks and use Effects.

For example, when you click on the View All button, it sets our showAllView hook to

true, which then causes the ShowAll component to render, and the Crud component to disappear. This causes a change in appearance where it look as though you have entered a whole new page:



After clicking View All:

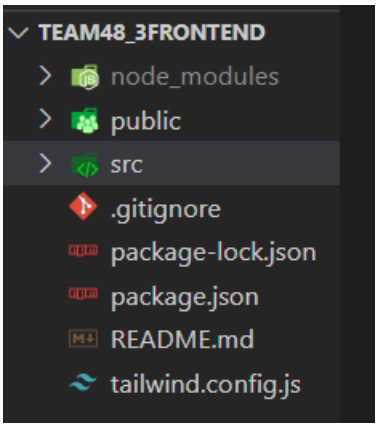


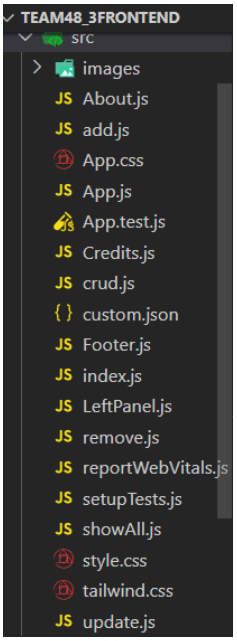
If I then were to click the Back button, it would bring me back to the first view.

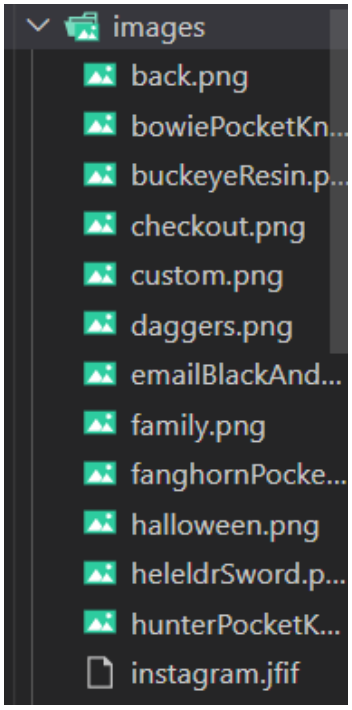
Directory Architecture

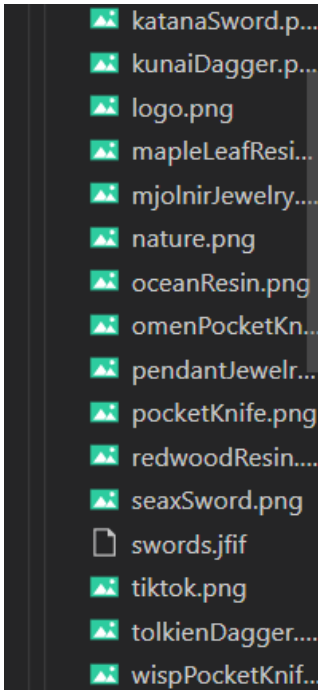
This project is divided into 2 subfolders: the frontend, which contains the user interface, and the backend, which creates a server where frontend requests are sent and data is pulled from our database to be read, created, updated, or deleted

Frontend structure:



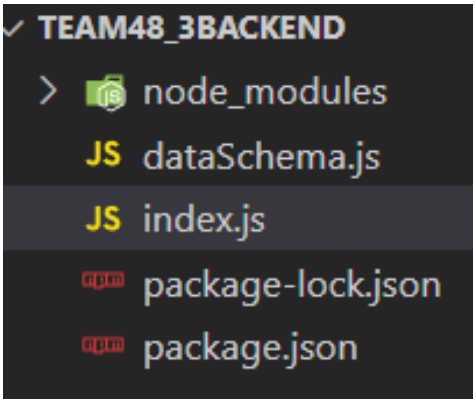






The bulk of the frontend code is contained in the crud.js, add.js, showAll.js, remove.js, and update.js code. These are each component that is called in the main driver, which is app.js.

Backend structure:



The backend code is comprised of index.js. This is the file where the server is created where and the database is accessed. This allows us to perform CRUD operations such as viewing, adding, updating, and deleting though our MongoDB database.

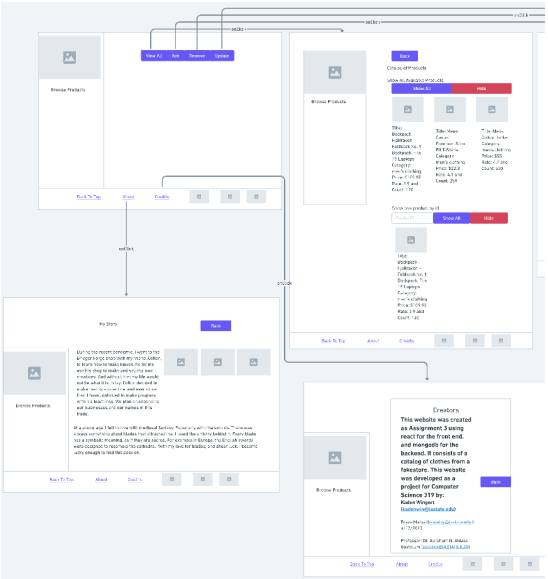
Server Architecture

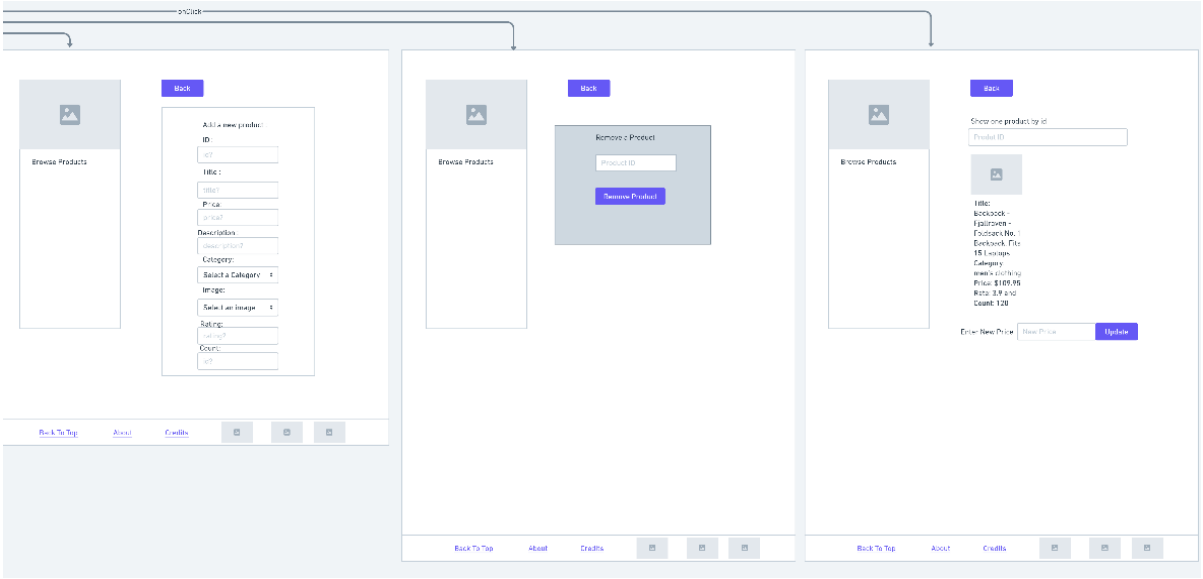
Our server is composed of an index.js file in the backend directory of our project. This file sets up our server using the Express.js framework as a localhost:3000 port and connects to a MongoDB database using the Mongoose library. It also defines endpoints for performing CRUD (Create, Read, Update, and Delete) operations on a collection of products stored in the database. The first few lines of code import dependencies such as Cors, Express.js, and Mongoose, . Then, we create an instance of the Express.js application and set it to a constant called app.

The code then uses middleware functions through app.use(). The express.json() middleware parses incoming requests with JSON payloads. The cors() middleware function enables cors so that the server can handle requests from different origins. The code connects to a MongoDB database using Mongoose's connect() method. It specifies the name of the database, the connection URL, and some connection options. We then define endpoints for CRUD operations on the product collection. The app.get() function defines an endpoint that retrieves all the products in the collection and sends them as a response. Another app.get() function defines an endpoint that retrieves a single product by its ID. The app.post() function defines an endpoint for creating a new product. It first checks if the product with the specified ID already exists in the collection, and if it does, it sends an error response. If the product does not exist, it

creates a new instance of the Product model, defined in the dataSchema.js file, with the data from the request body and saves it to the database. The app.put() function defines an endpoint for updating an existing product. It takes the product ID from the URL parameter and the new data from the request body. It uses Mongoose's findOneAndUpdate() method to find the product by its ID, update it with the new data, and return the updated product as a response. The app.delete() function defines an endpoint for deleting an existing product. It takes the product ID from the URL parameter, checks if the product exists, and, if it does, deletes it from the database using Mongoose's findByIdAndDelete() method. Finally, the code starts the server and listens for incoming requests on the specified port.

Logical Architecture





These images were created using Figma to show the flow of our website as you click different buttons to navigate through different views. Alternatively, this wireframe can be accessed through this link:
https://whimsical.com/finalproject-2HA94VaQ83mhfipaHaaS8Z

Explanation of each view

1. ViewAll: When this page loads, it uses the GET method to request all items from the server, which fetches them from the MongoDB database. The search functionality uses the GET method as well, but the request includes the item ID as a parameter.
2. Add: This component uses a form to collect details about a new item. When the form is submitted, it uses the POST method to send the item's details to the server. The server then creates a new item in the MongoDB database with these details.
3. Remove: This component accepts an item ID from the user. When the user confirms they want to delete the item, it uses the DELETE method to send a request to the server, including the item ID. The server then deletes the corresponding item from the MongoDB database.
4. Update: This component lets users find an item by ID (using a GET request) and update the item's price. When the user submits the new price, it uses the PUT method to send the updated price to the server, which then updates the item in the MongoDB database.
5. Credits: This component shows the names and emails of the creators of this website. It also shows which class it was made for, as well as the course's instructor.
6. About: This component displays images of the owner of Nordland Forge as well as several paragraphs explaining how he began this business.

Manual of Installation

Since we used many software applications, we needed to install them first. In the front end, we created a react project by running: npm create-react-app frontend then ran npm init.

Additionally, we used bootstrap and tailwind to style our products. This was done by: Running: npm install tailwindcss, then add the tailwind configuration file by typing npx tailwindcss init in the terminal. Finally, we added these lines to our styles.css:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

For bootstrap, we ran npm install bootstrap in our frontend directory. Then we import, tailwind,bootstrap, and our styles.css into our App.js file:

```
import "./style.css";
import './tailwind.css';
import 'bootstrap/dist/css/bootstrap.min.css';
```

For the backend, we used software such as node, express and Mongodb.

To install these, we ran npm init, and then:

Npm install express

Npm install cors

Npm install mongoose

Then, we use them by creating const variables in our backend index.js and set up middleware functions through app.use. The express.json() middleware is used to parse incoming requests with JSON format, and the cors() enable Cross-Origin Resource Sharing (CORS) so that the server can handle requests from different origins.

```
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());
mongoose.connect("mongodb://127.0.0.1:27017/reactdata", {
```

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

The code then connects to a MongoDB database using Mongoose's connect() method. It specifies the name of the database, the connection URL, and some connection options.

```
app.delete("/remove/:id", async (req, res) => {
```

Copy of the Code

Backend

Index.js

```
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,
});
const port = process.env.PORT || 4000;
const host = "localhost";

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.post("/add", async (req, res) => {
 console.log(req.body);
 const p_id = Number(req.body._id); // Convert to Number
 const productExists = await Product.findById(p_id);
 if (productExists) {
   return res.status(400).json({ message: `Product ${p_id} already exists` });
 }
 const ptitle = req.body.title;
 const pprice = req.body.price;
 const pdescription = req.body.description;
 const pcategory = req.body.category;
 const pimage = req.body.image;
 const prate = req.body.rating.rate;
 const pcount = req.body.rating.count;

 const formData = new Product({
   _id: p_id,
   title: ptitle,
   price: pprice,
   description: pdescription,
   category: pcategory,
   image: pimage,
   rating: { rate: prate, count: pcount },
 });

 try {
   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.put("/update/:id", async (req, resp) => {
 const id = req.params.id;
 const query = { _id: id };
 const update = {
   $set: req.body
 };
 const options = {
   new: true,
   useFindAndModify: false
 };
 const updatedProduct = await Product.findOneAndUpdate(query, update, options);
 console.log(updatedProduct);
 if (updatedProduct) {
   resp.send(updatedProduct);
 } else {
   resp.status(404).send({ message: `Product with id ${id} not found` });
 }
});
```

```
app.listen(port, () => {
 console.log(`App listening at http://%s:%s`, host, port);
});


app.delete("/remove/:id", async (req, res) => {
 const id = req.params.id;
 const productExists = await Product.findById(id);
 if (!productExists) {
   return res.status(404).json({ message: `Product ${id} does not exist` });
 }
 try {
   const removedProduct = await Product.findByIdAndDelete(id);
   if (!removedProduct) {
     return res.status(404).json({ message: `Product ${id} not found` });
   }
   res.json({ message: `Product ${id} removed` });
 } catch (err) {
   console.error(err);
   res.status(500).json({ message: 'Server error' });
 }
});
```

Frontend

App.js

```
import "./App.css";
import "./style.css";
import './tailwind.css';
import 'bootstrap/dist/css/bootstrap.min.css';

import React, { useState } from "react";
import Crud from "./crud";
import About from "./About";
import Credits from "./Credits";
import Footer from "./Footer";
import LeftPanel from "./LeftPanel";
import ShowAll from "./showAll";
import Add from "./add";
import Remove from "./remove";
import Update from "./update";

export const App = () => {

 const [showFooter, setShowFooter] = useState(true); //Footer doesn't appear on
confirmation
 const [showAbout, setShowAbout] = useState(false); //About page
 const [showCredits, setShowCredits] = useState(false); //Credits page
 const [isCrudVisable, setIsCrudVisable] = useState(true); //crud buttons
 const [showAllView, setShowAllView] = useState(false); //show all button
 const [showAddView, setShowAddView] = useState(false); //Add button
 const [showRemoveView, setShowRemoveView] = useState(false); //remove button
 const [showUpdateView, setShowUpdateView] = useState(false); //update button
```

```
const [isCrudBackVisable, setCrudBackVisable] = useState(false); //crud back button

const [product, setProduct] = useState([]);
const [viewer1, setViewer1] = useState(false);
const [oneProduct, setOneProduct] = useState([]);
const [viewer2, setViewer2] = useState(false);

const render_products = () => {
  return (
    <>
      {showAbout && (
        <About
          showAbout={showAbout}
          setShowAbout={setShowAbout}
          setIsCrudVisable={setIsCrudVisable}
        />
      )}
      {showCredits && (
        <Credits
          setShowCredits={setShowCredits}
          setIsCrudVisable={setIsCrudVisable}
        />
      )}

      {
        <Crud
          isCrudVisable={isCrudVisable}
          setIsCrudVisable={setIsCrudVisable}
          setShowAllView={setShowAllView}
          setShowAddView={setShowAddView}
          setShowRemoveView={setShowRemoveView}
          setShowUpdateView={setShowUpdateView}
          setCrudBackVisable={setCrudBackVisable}
        />
      }
      {isCrudBackVisable && (
        <>
          <button
            key="crudBackButton"
            className="crudButtons"
            onClick={() => {
              setCrudBackVisable(false);
              setIsCrudVisable(true);
              setShowAllView(false);
              setShowAddView(false);
              setShowRemoveView(false);
              setShowUpdateView(false);
            }}
          >
            Back
          </button>
        </>
      )}
      {
        <ShowAll
          showAllView={showAllView}
          isCrudBackVisable={isCrudBackVisable}
          setCrudBackVisable={setCrudBackVisable}
          product={product}
          setProduct={setProduct}
          viewer1={viewer1}
          setViewer1={setViewer1}
          oneProduct={oneProduct}
```

```
          setOneProduct={setOneProduct}
          viewer2={viewer2}
          setViewer2={setViewer2}
        />
      }
      {
        <Add
          showAddView={showAddView}
          isCrudBackVisable={isCrudBackVisable}
        />
      }
      {
        <Remove
          showRemoveView={showRemoveView}
          isCrudBackVisable={isCrudBackVisable}
        />
      }
      {
        <Update
        showUpdateView = {showUpdateView}
        isCrudBackVisable = {isCrudBackVisable}
        oneProduct = {oneProduct}
        setOneProduct = {setOneProduct}
        viewer2 = {viewer2}
        setViewer2 = {setViewer2}
        />
      }
    </>
  );
 };

 return (
   <div className="flex min-h-screen flex-row" style={{ height: "56em",
backgroundColor:"burlywood" }}>
     {
       <LeftPanel/>
     }
     <div className="ml-5 p-10 xl:basis-4/5">
       {render_products()}
     </div>
     {
       <Footer
         setShowAbout={setShowAbout}
         setShowCredits={setShowCredits}
         showFooter={showFooter}
         setShowFooter={setShowFooter}
       />
     }
   </div>
 );
}; //end App
```

Crud.js

```
import React from "react";

function Crud({
 isCrudVisable,
 setIsCrudVisable,
```

```
 setShowAllView,
 setShowAddView,
 setShowRemoveView,
 setShowUpdateView,
 setCrudBackVisable,
}) {
 return (
   <div>
     {isCrudVisable && (
       <div
         className="btn-group btn-group-lg d-flex"
         style = {{display: "flex", flexDirection:"column"}}
         role="group"
       >
         <button
           key="All"
           className="btn btn-primary btn-lg btn-block"
           style = {{borderBottomLeftRadius:"0px", borderTopLeftRadius:"0px",
marginLeft:"-1px"}}
           onClick={() => {
             setShowAllView(true);
             setCrudBackVisable(true);
             setIsCrudVisable(false);
           }}
         >
           View All
         </button>

         <button
           key="Add"
           className="btn btn-primary btn-lg btn-block"
           onClick={() => {
             setShowAddView(true);
             setCrudBackVisable(true);
             setIsCrudVisable(false);
           }}
         >
           Add
         </button>

         <button
           key="Remove"
           className="btn btn-primary btn-lg btn-block"
           onClick={() => {
             setShowRemoveView(true);
             setCrudBackVisable(true);
             setIsCrudVisable(false);
           }}
         >
           Remove
         </button>

         <button
           key="Update"
           className="btn btn-primary btn-lg btn-block"
           style = {{borderTopRightRadius:"0px", borderBottomRightRadius:"0px" }}
           onClick={() => {
             setShowUpdateView(true);
             setCrudBackVisable(true);
             setIsCrudVisable(false);
           }}
         >
           Update
```

```
          </button>
        </div>
      )}
    </div>
  );
}

export default Crud;
```

add.js

```
import React, { useState } from "react";

function Add({ showAddView, isCrudBackVisable }) {
  const categoryImages = {
    "Pocket Knives": [
      require("./images/fanghornPocketKnife.png"),
      require("./images/hunterPocketKnife.png"),
      require("./images/wispPocketKnife.png"),
      require("./images/bowiePocketKnife.png"),
      require("./images/omenPocketKnife.png"),
    ],
    Daggers: [require("./images/tolkienDagger.png"),
require("./images/kunaiDagger.png")],
    Swords: [require("./images/heleldrSword.png"), require("./images/seaxSword.png"),
require("./images/katanaSword.png")],
    Resin: [
      require("./images/mapleLeafResin.png"),
      require("./images/redwoodResin.png"),
      require("./images/oceanResin.png"),
      require("./images/buckeyeResin.png"),
    ],
    Jewelry: [require("./images/mjolnirJewelry.png"),
require("./images/pendantJewelry.png") ],
    Custom: [require("./images/custom.png")],
  };

  const [selectedCategory, setSelectedCategory] = useState("");

  const handleCategoryChange = (e) => {
    setSelectedCategory(e.target.value);
  };

  const [addNewProduct, setAddNewProduct] = useState({
    _id: "",
    title: "",
    price: "",
    description: "",
    category: "",
    image: "",
    rating: { rate: "", count: "" },
  });

  function handleChange(evt) {
    const value = evt.target.value;
    if (evt.target.name === "_id") {
      setAddNewProduct({ ...addNewProduct, _id: parseInt(Math.abs(value)) });
    } else if (evt.target.name === "title") {
      setAddNewProduct({ ...addNewProduct, title: value });
    } else if (evt.target.name === "price") {
```

```
    setAddNewProduct({ ...addNewProduct, price: Math.abs(value) });
  } else if (evt.target.name === "description") {
    setAddNewProduct({ ...addNewProduct, description: value });
  } else if (evt.target.name === "category") {
    setAddNewProduct({ ...addNewProduct, category: value });
  } else if (evt.target.name === "image") {
    const temp = value;
    setAddNewProduct({ ...addNewProduct, image: temp });
  } else if (evt.target.name === "rate") {
    setAddNewProduct({
      ...addNewProduct,
      rating: { rate: Math.abs(value) },
    });
  } else if (evt.target.name === "count") {
    const temp = addNewProduct.rating.rate;
    setAddNewProduct({
      ...addNewProduct,
      rating: { rate: temp, count: Math.abs(value) },
    });
  }
}

function handleOnSubmit(e) {
  e.preventDefault();
  console.log(e.target.value);
  fetch("http://localhost:4000/add", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(addNewProduct),
  })
    .then((response) => {
      if (!response.ok) {
        throw response;
      }
      return response.json();
    })
    .then((data) => {
      console.log("Post a new product completed");
      console.log(data);
      if (data) {
        //const keys = Object.keys(data);
        const value = Object.values(data);
        alert(value);
      }
    })
    .catch((error) => {
      error.json().then((errorMessage) => {
        alert(errorMessage.message);
      });
    });
}

return (
  <>
    {isCrudBackVisable && showAddView && (
      <div className="form">
        <h3 className="motto">Add a new product :</h3>
        <form>
<div className="form-group">
  <label>ID</label>
  <input type="number" placeholder="id?" name="_id" value={addNewProduct._id}
onChange={handleChange} className="form-control" />
</div>
```

```
 <div className="form-group">
   <label>Title</label>
   <input type="text" placeholder="title?" name="title" value={addNewProduct.title}
onChange={handleChange} className="form-control" />
 </div>
 <div className="form-group">
   <label>Price</label>
   <input type="number" placeholder="price?" name="price" value={addNewProduct.price}
onChange={handleChange} className="form-control" />
 </div>
 <div className="form-group">
   <label>Description</label>
   <input type="text" placeholder="description?" name="description"
value={addNewProduct.description} onChange={handleChange} className="form-control" />
 </div>
 <div className="form-group">
   <label>Category</label>
   <select name="category" id="category" onChange={handleCategoryChange}
className="form-control">
     <option value="">--Select Category--</option>
     <option value="Pocket Knives">Pocket Knives</option>
     <option value="Daggers">Daggers</option>
     <option value="Swords">Swords</option>
     <option value="Resin">Resin</option>
     <option value="Jewelry">Jewelry</option>
     <option value="Custom">Custom</option>
   </select>
 </div>
 <div className="form-group">
   <label>Image</label>
   <select name="image" required onChange={handleChange} className="form-control">
     <option value="">Select an image</option>
     {selectedCategory && categoryImages[selectedCategory].map((image) => (
       <option key={image} value={image}>{image}</option>
     ))}
   </select>
 </div>
 <div className="form-group">
   <label>Rating</label>
   <input type="number" placeholder="rate?" name="rate"
value={addNewProduct.rating.rate} onChange={handleChange} className="form-control" />
 </div>
 <div className="form-group">
   <label>Count</label>
   <input type="number" placeholder="count?" name="count"
value={addNewProduct.rating.count} onChange={handleChange} className="form-control" />
 </div>
 <button type="submit" onClick={handleOnSubmit}
className="removeProductButton">submit</button>
</form>
       </div>
     )}
   </>
 );
}

export default Add;
```

Remove.js

```
import React, { useState } from "react";

function Remove({
 showRemoveView,
 isCrudBackVisable,
}) {
 const [productId, setProductId] = useState("");

 const handleRemoveProduct = () => {
   fetch(`http://localhost:4000/remove/${productId}`, {
     method: "DELETE",
   })
     .then((response) => {
       if (!response.ok) {
         throw response;
       }
       return response.json();
     })
     .then((data) => {
       console.log(data);
       alert(`Product ${productId} removed`);
     })
     .catch((error) => {
       error.json().then((errorMessage) => {
         alert(errorMessage.message);
       });
     });
 };

 return (
   <>
     {isCrudBackVisable && showRemoveView && (
       <>
       <div className="removeProductSection">
         <div>
           <h3 className="removeProductTitle">Remove a product:</h3>
           <input
           className="removeProductInput"
           type="text"
           placeholder="Product ID"
           value={productId}
           onChange={(e) => setProductId(e.target.value)}
         />
           <button className="removeProductButton" onClick={handleRemoveProduct}>
           Remove Product
         </button>
         </div>
         </div>
       </>
     )}
   </>
 );
}

export default Remove;
```

Update.js

```
import React, { useState } from "react";

function Update({
 showUpdateView,
 isCrudBackVisable,
 oneProduct,
 setOneProduct,
 viewer2,
 setViewer2,
}) {
 const [shouldRefresh, setShouldRefresh] = useState(false);

 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);
       });
     setViewer2(!viewer2);
   } else {
     console.log("Wrong number of Product id.");
   }
 }
 async function updatePrice(id) {
   if (id === "") {
     alert("Must have a valid ID in the input");
     return;
   }
   const newPrice = Math.abs(
     parseFloat(document.getElementById("newPrice").value)
   );
   console.log(
     "IN UPDATE PRICE, Element's ID: ",
     document.getElementById("message").value
   );
   console.log(
     "IN UPDATE PRICE, New price: ",
     document.getElementById("newPrice").value
   );
   setShouldRefresh(true);
   await fetch("http://localhost:4000/update/" + id, {
     method: "PUT",
     headers: { "content-type": "application/json" },
     body: JSON.stringify({
       price: newPrice,
     }),
   })
     .then((response) => {
       if (response.ok) {
         console.log("Fetch works");
         return response.json();
       } else {
         throw response;
       }
     })
```

```
    .then((data) => {
      console.log(data);
      // Update the state with the updated product
      setOneProduct([data]);
    })
    .then(() => {
      setShouldRefresh(true);
    })
    .catch((error) => {
      error.text().then((errorMessage) => {
        alert(errorMessage);
      });
    });
}

const showOneItem = oneProduct.map((el) => (
  <div key={el._id} className="col-sm-12 col-md-6 col-lg-4 mb-4">
    <img src={el.image} width={150} className="img-fluid img-thumbnail" />{" "}
    <br />
    <span className="fw-bold">Title:</span> {el.title} <br />
    <span className="fw-bold">Category:</span> {el.category} <br />
    <span className="fw-bold">Price:</span> ${el.price} <br />
    <span className="fw-bold">Rate:</span> {el.rating.rate} and{" "}
    <span className="fw-bold">Count:</span> {el.rating.count} <br />
  </div>
));

return (
  <>
    {isCrudBackVisable && showUpdateView && (
      <div
        className="viewAllProducts crud container-fluid p-5"
        style={{ backgroundColor: "burlywood" }}
      >
        <h1 className="h1">Catalog of Products </h1>
        <br />
        <div className="oneProductContainer">
          <h1 className="oneProduct">Show one Product by Id:</h1>
          <div className="input-group mb-3">
            <input
              type="text"
              id="message"
              name="message"
              placeholder="ID"
              className="form-control"
              onChange={(e) => getOneProduct(e.target.value)}
            />
            <button
              className="btn btn-sm btn-primary"
              type="button"
              id="button-addon2"
              onClick={() =>
                getOneProduct(document.getElementById("message").value)
              }
            >
              Show
            </button>
          </div>

          {viewer2 && <div className="row products">{showOneItem}</div>}
          {viewer2 && (
            <div className="input-group mb-3">
              <div
```

```
                  style={{
                    display: "flex",
                    flexDirection: "column",
                    width: "100%",
                  }}
                >
                  <h1 className="oneProduct">Enter new Price:</h1>
                  <div style={{ display: "flex", alignItems: "center" }}>
                    <input
                      type="text"
                      id="newPrice"
                      name="message"
                      placeholder="new price"
                      className="form-control"
                      style={{ width: "100%" }}
                    />
                    <button
                      className="btn btn-md btn-primary"
                      type="button"
                      id="button-addon2"
                      style = {{marginLeft:"-2px",
borderTopLeftRadius:"0px",borderBottomLeftRadius:"0px"}}
                      onClick={(e) => {
                        console.log(
                          "Element's ID: ",
                          document.getElementById("message").value
                        );
                        updatePrice(document.getElementById("message").value);
                      }}
                    >
                      Update
                    </button>
                  </div>
                </div>
              </div>
            )}
          </div>
          <hr />
        </div>
      )}
    </>
  );
}

export default Update;
```

ShowAll.js

```
import React, { useEffect, useState } from "react";
function ShowAll({
 showAllView,
 isCrudBackVisable,
 product,
 setProduct,
 viewer1,
 setViewer1,
 oneProduct,
 setOneProduct,
 viewer2,
 setViewer2,
```

```
}) {
 const [shouldRefetch, setShouldRefetch] = useState(false);

 useEffect(() => {
   if (shouldRefetch) {
     getAllProducts();
     setShouldRefetch(false);
   }
 }, [shouldRefetch]);

 function getAllProducts() {
   fetch("http://localhost:4000/")
     .then((response) => response.json())
     .then((data) => {
       console.log("Show Catalog of Products :");
       console.log(data);
       setProduct(data);
       setViewer1(true); // Show the component after data is fetched
     });
 }


 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);
         setViewer2(true); // Show the component after data is fetched
       });
   } else {
     console.log("Wrong number of Product id.");
   }
 }
 const showAllItems = product.map((el) => (
   <div key={el._id} className="col-sm-12 col-md-6 col-lg-4 mb-4">
     <img src={el.image} width={150} className="img-fluid img-thumbnail"/> <br />
     <span className="fw-bold">Id:</span> {el._id} <br />
     <span className="fw-bold">Title:</span> {el.title} <br />
     <span className="fw-bold">Category:</span> {el.category} <br />
     <span className="fw-bold">Price:</span> ${el.price} <br />
     <span className="fw-bold">Rate:</span> {el.rating.rate} and <span
className="fw-bold">Count:</span> {el.rating.count} <br />
   </div>
 ));
  const showOneItem = oneProduct.map((el) => (
   <div key={el._id} className="col-sm-12 col-md-6 col-lg-4 mb-4">
     <img src={el.image} width={150} className="img-fluid img-thumbnail" /> <br />
     <span className="fw-bold">Id:</span> {el._id} <br />
     <span className="fw-bold">Title:</span> {el.title} <br />
     <span className="fw-bold">Category:</span> {el.category} <br />
     <span className="fw-bold">Price:</span> ${el.price} <br />
     <span className="fw-bold">Rate:</span> {el.rating.rate} and <span
className="fw-bold">Count:</span> {el.rating.count} <br />
   </div>
 ));
 return (
   <>
```

```
{isCrudBackVisable && showAllView && (
 <div className="viewAllProducts crud container-fluid p-5" style={{ backgroundColor:
"burlywood" }}>
   <h1 className="catalogOfProducts mb-5">Catalog of Products</h1>
   <div className="row justify-content-between align-items-center mb-5">
     <h2 className="showAllavailable">Show All Available Products:</h2>
     <div className="btn-group" role="group" aria-label="Basic example">
 <button type="button" className="btn btn-sm btn-primary" onClick={() =>
getAllProducts()}>Show All</button>
 <button type="button" className="btn btn-sm btn-danger" onClick={() =>
setViewer1(false)}>Hide</button>
</div>

   </div>
   <hr />
   {viewer1 && <div className="row products">{showAllItems}</div>}
   <hr />
   <div className="oneProductContainer">
     <h2 className="oneProduct">Show One Product by ID:</h2>
     <div className="input-group mb-3">
       <input
         type="text"
         id="productId"
         className="form-control"
         placeholder="Product ID"
         aria-label="Product ID"
         aria-describedby="button-addon2"
         onChange={(e) => getOneProduct(e.target.value)}
       />
       <button
         className="btn btn-sm btn-primary"
         type="button"
         id="button-addon2"
         onClick={() => getOneProduct(document.getElementById("productId").value)}
       >
         Show
       </button>
       <button className="btn btn-sm btn-danger" onClick={() =>
setViewer2(false)}>Hide</button>
     </div>
     {viewer2 && <div className="row products">{showOneItem}</div>}
   </div>
   <hr />
 </div>
)}
   </>
 );
}
export default ShowAll;
```