SE/COMS 319 Construction of User Interfaces

Dr. Aldaco

Spring 2024

Final Project Documentation

Scooter's Pet Shop

Team 77

Justin Sebahar

Carter Peterson

5/6/24

Table of Contents

Project Description

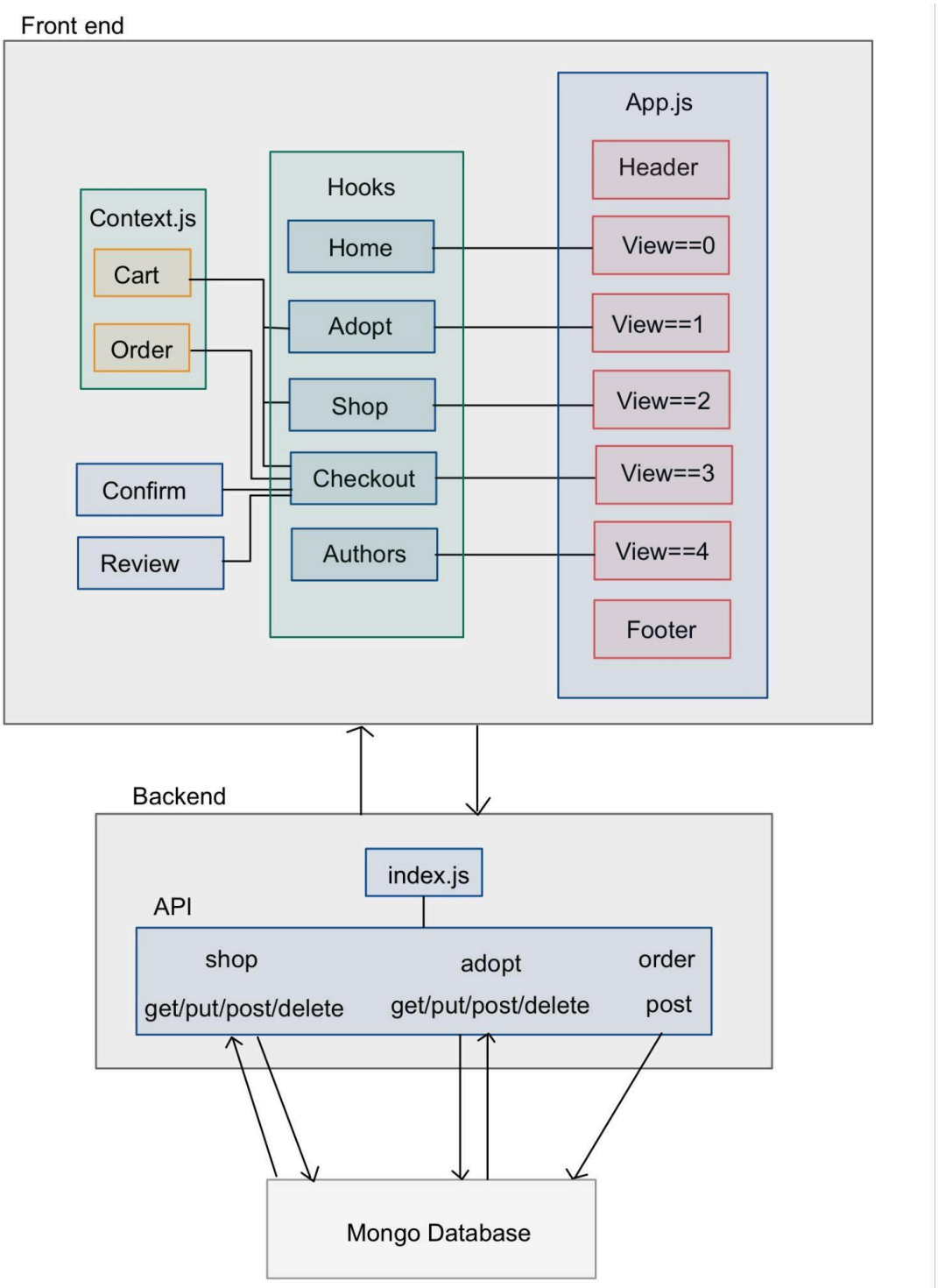
Like most people, we love animals! From dogs to turtles, pets play an important role in the lives of many. That was the inspiration to create a mock pet shop website for our midterm and final projects. Since the start of the project, things have come a long way; we hope you enjoy our work!

The aim of our website is to simulate what you may experience on a typical pet shop's website. The over-arching goal is for the user to browse a selection of products for sale or a list of animals in the shelter needing adoption. User can add products to their cart and "purchase" them. Furthermore, a user can select from the list of animals, one that they may wish to "reserve" (expressing interest in meeting and/or adopting an animal.

While building our project, we had an additional goal in mind of creating the most aesthetically pleasing, yet simplistic and effective design we could. Many pet store websites we looked at (even large ones) were very cluttered, confusing, and boring. Bearing that in mind, we created what we think is an effective yet visually pleasing website.

Software/Logical Architecture

Software Diagram



The user starts on the home page. From there they may navigate to the shop or adoption page. On the shop page, they can browse items fetched from the “shop” collection/API. These items may be added or removed from the cart to whatever quantity desired (assuming they are not out of stock). Likewise, on the adoption page, pets are fetched and listed that are available for adoption. If a user shows interest in meeting an animal, they may “reserve” it by clicking the reserve button to add it to the cart (they can only reserve one animal, and only ones that are available (non-red button)).

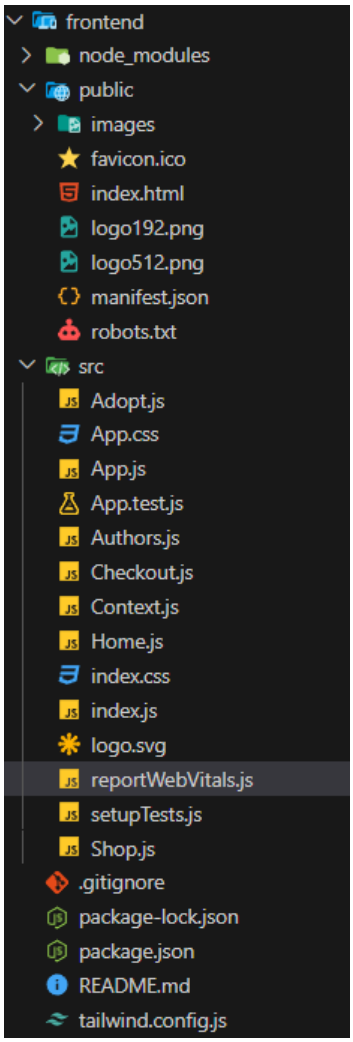
Continuing to the cart, all items (and potentially an animal) will be displayed with their details, including total prices. To confirm the transaction, the form must be properly filled out and a button placed. Upon the transaction:

- A post is made to the “order” collection with all form details and purchased items.
- For each item purchased, a put is made to decrease the stock by the quantity purchased.
- A put will be made for an animal reserved, marking it as so.

Finally, the receipt page will display all transaction details, such as items, prices, and form info. Additionally, the user can optionally submit a rating for any of the items they purchased. They can choose from 1-5 stars and upon submission, a put is made for that item, updating its rating.

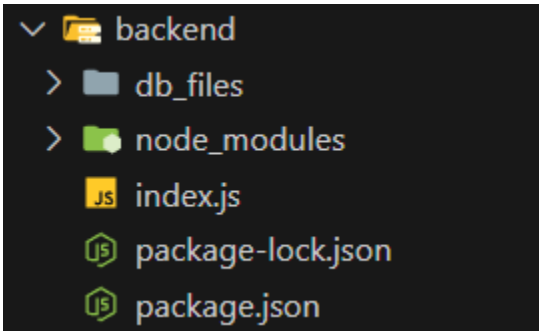
File Structure

Frontend: For the frontend (located in the frontend directory), each view gets its own js file, located within the src directory and exporting the unique hook. They are then all linked together in the App.js file, which handles transitioning between views, along with displaying the header and footer. Context.js allows the use of context between hooks to share the needed state variables. This keeps things better organized. Additionally, static embedded images/icons are stored in public/images.



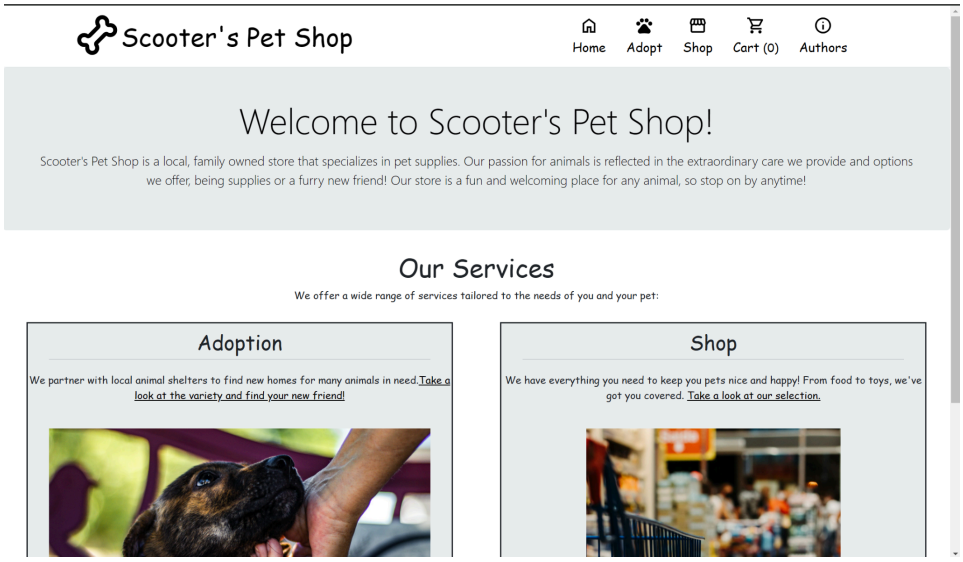
Backend: All of the backend (backend directory) APIs are done in index.js, which has everything for the Shop, Adopt, and Order APIs. Also, db_files contains a copy of the

data contained in our database; it is not used in the application, but rather there for convenience.



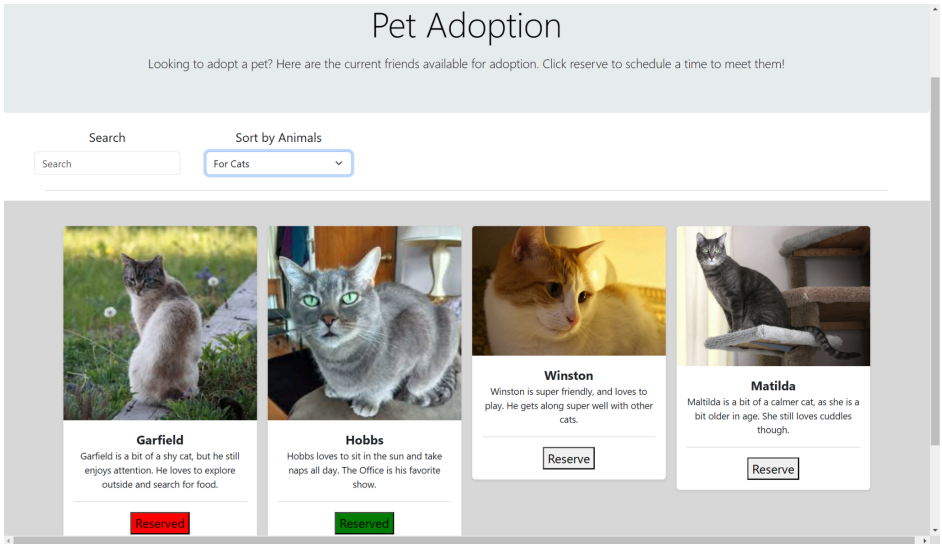
User Views

Home View:



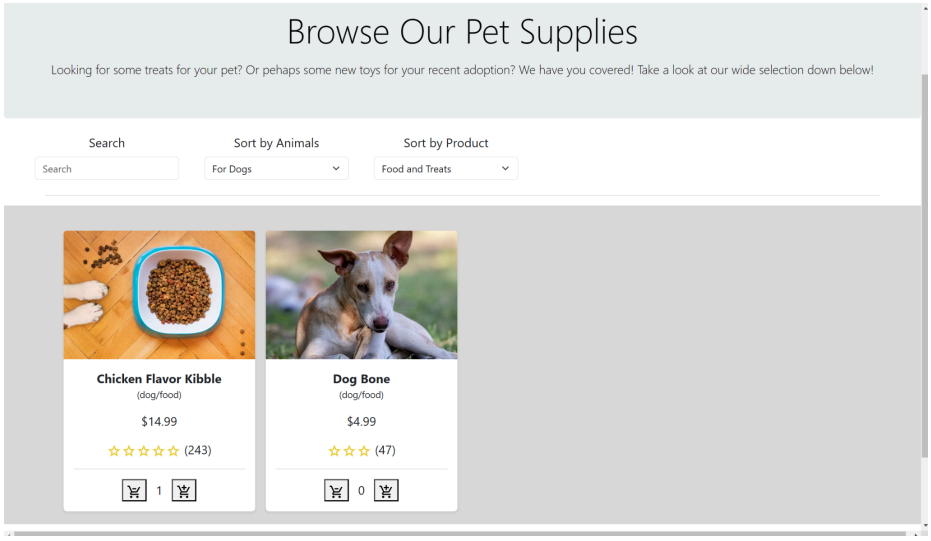
The home view has information about the pet shop, including the options to either shop for things or look for pets to adopt.

Adopt View:



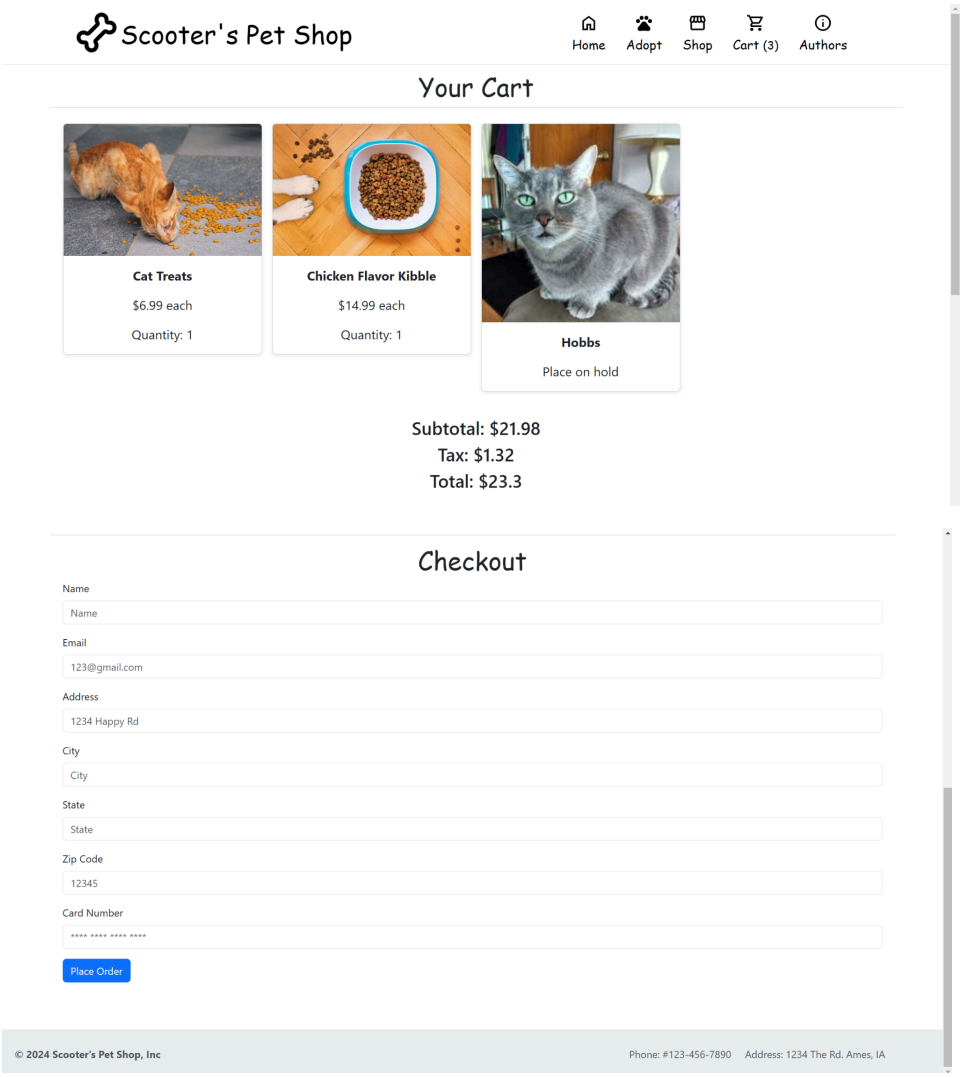
The adopt view shows current pets up for adoption. You can sort by cats or dogs, and also search by their name. When you click on the reserve button the pet is added to your cart, and the button appears green. If pets are already reserved by someone else, the reserved button is red. Each user can only reserve one pet at a time.

Shop View:



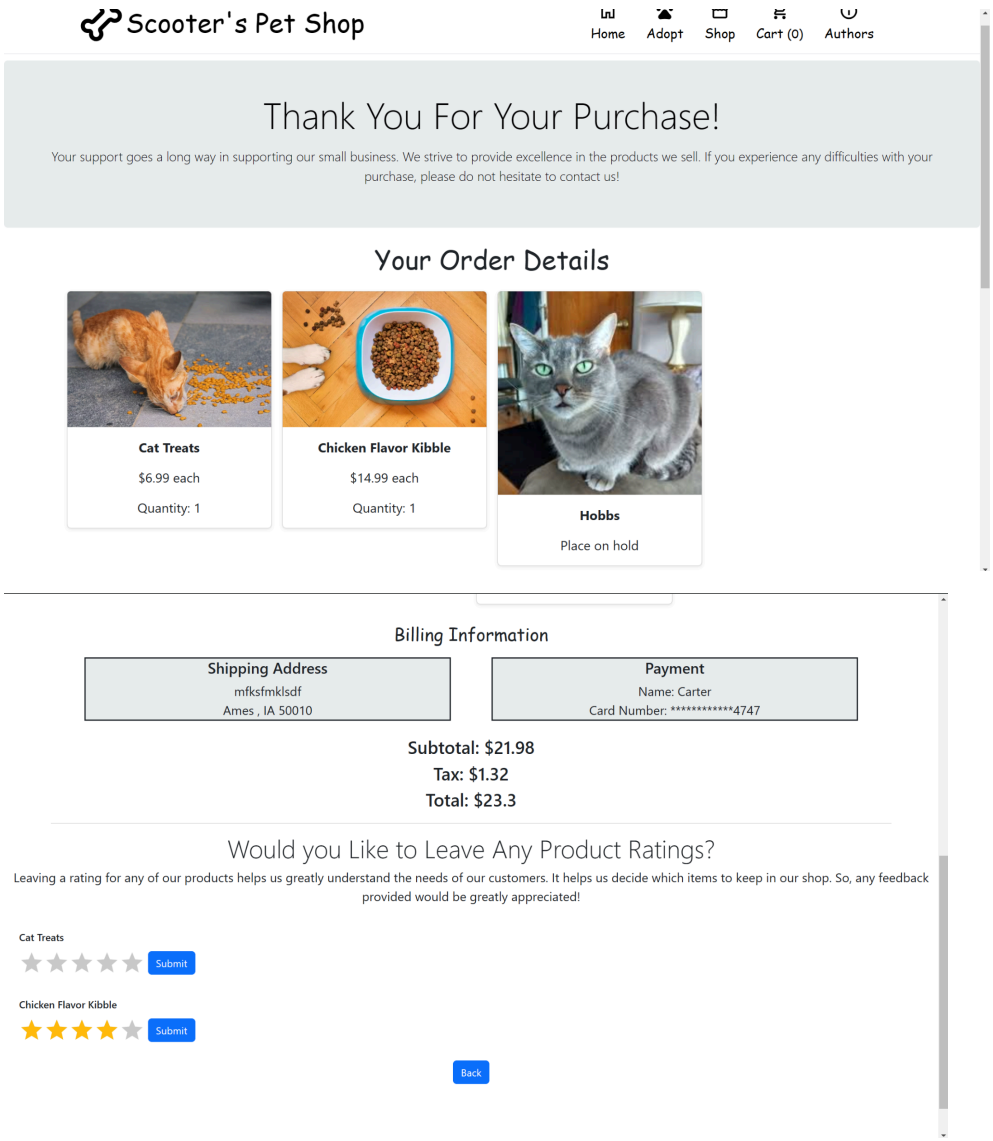
The shop view allows you to shop for pet supplies, toys, and treats. You can search by name, sort by animal, and sort by product type. When you see a product you are interested in, you can click the add or remove button to change how many of that product you would like to purchase. If that product is out of stock, you get alerted that no stock is available.

Cart View:



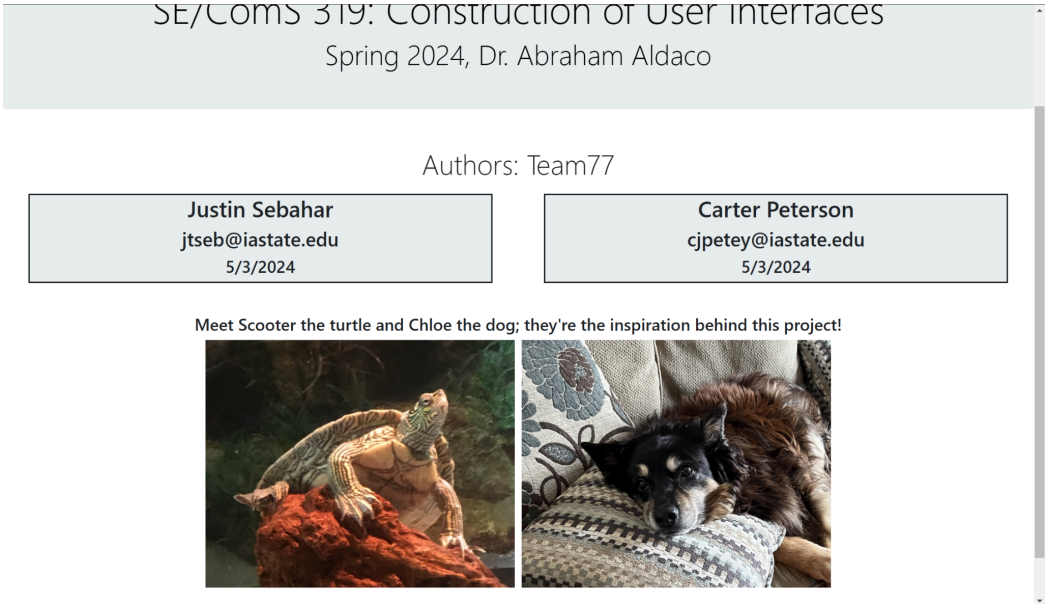
Cart view shows you the current items in your cart, in addition to the pets you have reserved to adopt. Below that is the checkout section, where you can enter information to proceed with the checkout. There is verification that your email address is a real email, and that your credit card is a real number using regular expressions.

Post Checkout View:



Post checkout view gives you a summary of what you just purchased, billing information, shipping address, and total. There is also a spot on the button where you can leave product reviews once you figure out if you like them or not.

Authors view:



The author view provides basic information about the project's authors, along with the inspiration for the project.

Installation

This project must be configured correctly and contain all required dependencies to function. Additionally, the application must be run using “npm run start” to load the front end and “nodemon index.js” to run the back end. The application was built using React and Node, so those must be installed in the system as well.

The following modules must be installed on the frontend:

- bootstrap
- @material-tailwind/react
- feedback
- list-item
- react-bootstrap
- react-listview
- react-simple-star-rating
- tailwindcss
- tailwind
- use-between
- use-context

For reference, the package.json for the frontend may be found on pg. 51, containing a detailed description of project dependencies.

The following module must be installed on the backend:

- body-parser
- cors
- express
- mongodb
- node
- nodemon

For reference, the package.json for the backend may be found on pg. 61, containing a detailed description of project dependencies.

Furthermore, the mongo database must be named SE319FINAL and contain the following collections:

- adopt; format for data:

```
{
  "id": "A",
  "animal": "cat",
  "name": "Garfield",
  "age": "5 years",
  "description": "Garfield is a bit of a shy cat, but he still enjoys attention. He loves to explore outside and search for food.",
  "image": "url,
  "isReserved": false
}
```

- Order (only posts are made to here)
- Shop; format for data:

```
{
  "id": "1",
  "type": "food",
  "animal": "dog",
  "title": "Biscuit Treats",
  "price": "$9.99",
  "image": "url,
  "rate": {
    "rating": "3.97",
    "count": "87"
   }
```

Code

Frontend

src/App.js

```
import "./App.css";
import React, { useState, useEffect } from "react";
import { Context } from "./Context";
import { useContext } from "react";
import Home from "./Home";
import Shop from "./Shop";
import Adopt from "./Adopt";
import Checkout from "./Checkout";
import Authors from "./Authors"

function App() {
  const [View, setView] = useState(0);
  const { cart, order } = useContext(Context);
  const [Cart, setCart] = cart;

  function toHome() {
    setView(0);
  }

  function toAdopt() {
    setView(1);
  }

  function toShop() {
    setView(2);
  }

  function toCart() {
    setView(3);
  }

  function toAuthors() {
    setView(4);
```

```
  }

  return (
    <div className="App">
      <header>
        <div className="px-3 py-2 border-bottom">
          <div className="container">
            <div className="d-flex flex-wrap align-items-center
justify-content-center justify-content-lg-start">
              <a
                style={{ paddingLeft: 15 + "px" }}
                href="./index.html"
                className="d-flex align-items-center my-2 my-lg-0
me-lg-auto text-white text-decoration-none"
              >
                <img
                  src="images/shop.svg"
                  className="bi d-block mx-auto mb-1"
                  width="75px"
                  height="75px"
                ></img>
                <span style={{ fontSize: 40 + "px" }}>Scooter's Pet
Shop</span>
              </a>

              <ul
                style={{ paddingRight: 50 + "px" }}
                className="nav col-12 col-lg-auto my-2
justify-content-center my-md-0 text-small"
              >
                <li>
                  <button onClick={toHome} className="nav-link
text-secondary">
                    <img
                      src="./images/home.svg"
                      className="bi d-block mx-auto mb-1"
                      width="30"
                      height="30"
                    ></img>
                    <span id="home" className="hover">
```

```
                    Home
                  </span>
                </button>
              </li>
              <li>
                <button onClick={toAdopt} className="nav-link
text-white">
                  <img
                    src="./images/adopt.svg"
                    className="bi d-block mx-auto mb-1"
                    width="30"
                    height="30"
                  ></img>
                  <span className="hover">Adopt</span>
                </button>
              </li>
              <li>
                <button onClick={toShop} className="nav-link
text-white">
                  <img
                    src="./images/storefront.svg"
                    className="bi d-block mx-auto mb-1"
                    width="30"
                    height="30"
                  ></img>
                  <span className="hover">Shop</span>
                </button>
              </li>
              <li>
                <button onClick={toCart} className="nav-link
text-white">
                  <img
                    src="./images/cart.svg"
                    className="bi d-block mx-auto mb-1"
                    width="30"
                    height="30"
                  ></img>
                  <span className="hover">
                    Cart{" "}
                    <span style={{ fontSize: 18 + "px" }}>
```

```
                    ({Cart.length})
                  </span>
                </span>
              </button>
            </li>
            <li>
              <button onClick={toAuthors} className="nav-link
text-white">
                <img
                  src="./images/info.svg"
                  className="bi d-block mx-auto mb-1"
                  width="30"
                  height="30"
                ></img>
                <span className="hover">Authors</span>
              </button>
            </li>
          </ul>
        </div>
      </div>
    </div>
  </header>

  <div>
    {View === 0 && <Home />}
    {View === 1 && <Adopt />}
    {View === 2 && <Shop />}
    {View === 3 && <Checkout />}
    {View === 4 && <Authors />}
  </div>
  <footer
    id="footer"
    className="d-flex flex-wrap justify-content-between
align-items-center border-top"
  >
    <p
      style={{ paddingLeft: 20 + "px" }}
      className="col-md-4 mb-0 text-body-secondary"
    >
      <strong>&copy; 2024 Scooter's Pet Shop, Inc</strong>
```

```
        </p>
        <p className="col-md-4 mb-0 text-body-secondary">
          Phone: #123-456-7890     Address: 1234 The Rd. Ames,
IA
        </p>
      </footer>
    </div>
  );
}

export default App;
```

src/Adopt.js

```
import React, { useContext } from "react";
import { useState, useEffect } from "react";
import { Context } from "./Context";

function Adopt() {
  const { cart, order } = useContext(Context);
  const [Cart, setCart] = cart;
  const [Pets, setPets] = useState([]);
  const [initialized, setInitialized] = useState(false);
  const [Results, setResults] = useState([]);
  const [Animal, setAnimal] = useState("all");
  const [Type, setType] = useState("all");

  useEffect(() => {
    fetch("http://localhost:8080/adopt")
      .then((response) => response.json())
      .then((pets) => {
        setPets(pets);
      });
  });

  // Use separate effect for Results of sort so it is not ovverriden by
updates to Products.
  useEffect(() => {
    if (initialized === false) {
```

```
    fetch("http://localhost:8080/adopt")
      .then((response) => response.json())
      .then((results) => {
        setResults(results);
        setInitialized(true);
      });
  }
});

useEffect(() => {
  handleSearch();
}, [Animal, Type]);

function addToCart(pet) {

  for (let i in Cart) {
    const product = Cart[i];
    if(product.id == pet.id){
      removeFromCart(pet);
      let button = document.getElementById(pet.id);
      button.style.backgroundColor = "white";
      button.innerHTML = "Reserve"
      return;
    }
    else if(product.title == null){
      alert("You may only reserve one pet at a time!");
      return;
    }
  }

    let copy = [...Cart, pet];
    console.log(copy);
    setCart(copy);
    let button = document.getElementById(pet.id);
    button.style.backgroundColor = "green";
    button.innerHTML = "Reserved"
}

const isReserved = (pet) => {
  for (let i in Cart) {
```

```
      const product = Cart[i];
      if(product.id == pet.id){
        return (
          <button id={pet.id} style={{ backgroundColor: "green"}}
className="reservedButton" onClick={() =>
addToCart(pet)}>Reserved</button>
      );
      }
    }
    if(pet.isReserved == false){
        return (
            <button id={pet.id} className="reservedButton" onClick={() =>
addToCart(pet)}>Reserve</button>
        );
    }
    else{
        return (
            <button id={pet.id} className="reservedButton" style={{
backgroundColor: "red"}}>Reserved</button>
        );
    }

  }

  function removeFromCart(product) {
    let cartCopy = [];
    let removeIndex = -1;
    for (let i = 0; i < Cart.length; i++) {
      if (Cart[i].id === product.id) {
        removeIndex = i;
        break;
      }
    }

    let copyInd = 0;
    for (let i = 0; i < Cart.length; i++) {
      if (i != removeIndex) {
        cartCopy[copyInd] = Cart[i];
        copyInd++;
      }
```

```
    }
    setCart(cartCopy);
    console.log(Cart);
  }

  function handleSearch() {
    var value = document.getElementById("search").value;
    const results = Pets.filter((p) => {
      if (Animal === "all" && Type === "all") {
        return p.name.toLowerCase().includes(value.toLowerCase());
      } else if (Animal === "all" && Type !== "all") {
        return (
          p.name.toLowerCase().includes(value.toLowerCase()) &&
          p.type.toLowerCase() === Type
        );
      } else if (Animal !== "all" && Type === "all") {
        return (
          p.name.toLowerCase().includes(value.toLowerCase()) &&
          p.animal.toLowerCase() === Animal
        );
      } else {
        return (
          p.name.toLowerCase().includes(value.toLowerCase()) &&
          p.animal.toLowerCase() === Animal &&
          p.type.toLowerCase() === Type
        );
      }
    });
    console.log(results);
    setResults(results);
  }

  return (
    <div>
      <div
        id="main"
        className="p-4 p-md-5 mb-4 rounded text-body-emphasis
bg-body-secondary"
      >
        <h1 className="display-4">Pet Adoption</h1>
```

```
        <p className="lead my-3">
          Looking to adopt a pet? Here are the current friends available
for adoption. Click reserve to
          schedule a time to meet them!
        </p>
      </div>

      <div
        style={{ textAlign: "center", fontSize: 20 + "px" }}
        className="row g-3"
      >
        <div
          style={{ width: 250 + "px", marginLeft: 50 + "px" }}
          className="col-md-5"
        >
          <label htmlFor="search" className="form-label">
            Search
          </label>
          <input
            type="text"
            className="form-control"
            id="search"
            placeholder="Search"
            onChange={handleSearch}
          />
        </div>

        <div
          style={{ width: 250 + "px", marginLeft: 25 + "px",
paddingBottom: 25 + "px" }}
          className="col-md-5"
        >
          <label htmlFor="animal" className="form-label">
            Sort by Animals
          </label>
          <select
            onChange={(e) => {
              setAnimal(e.target.value);
            }}
            className="form-select"
```

```
            id="animal"
          >
            <option value="all">All</option>
            <option value="cat">For Cats</option>
            <option value="dog">For Dogs</option>
          </select>
          <div className="invalid-feedback">Please make a valid
selection.</div>
        </div>

        <hr style={{ marginTop: 25 + "px", width: 90 + "%", margin: "auto"
}} />

        <div style={{ backgroundColor: "#d9d9d9", paddingTop: 40 + "px"
}}>
          <div className="container">
            <div
              className="row row-cols-1 row-cols-sm-2 row-cols-md-3
row-cols-lg-4 g-3"
              style={{ paddingBottom: 20 + "px" }}
            >
              {Results.map((pet) => (
                <div key={pet.id} className="col">
                  <div className="card shadow-sm">
                    <img
                      src={pet.image}
                      className="card-img-top"
                      alt="Item Image"
                    ></img>
                    <div className="card-body">
                      <p className="card-text" style={{ fontSize: 15 +
"px" }}>

                        {" "}
                        <strong style={{ fontSize: 20 + "px" }}>
                          {pet.name}
                        </strong>
                        <br />{pet.description}
                      </p>
                      <hr />
                      <div>
```

```
                        {isReserved(pet)}
                      </div>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          </div>
        </div>
      </div>
    );
}

export default Adopt;
```

src/Shop.js

```
import React, { useContext } from "react";
import { useState, useEffect } from "react";
import { Context } from "./Context";

function Shop() {
  const {cart, order} = useContext(Context);;
  const [Cart, setCart] = cart;
  const [Products, setProducts] = useState([]);
  const [initialized, setInitialized] = useState(false);
  const [Results, setResults] = useState([]);
  const [Animal, setAnimal] = useState("all");
  const [Type, setType] = useState("all");

  useEffect(() => {
    fetch("http://localhost:8080/shop")
      .then((response) => response.json())
      .then((products) => {
        setProducts(products);
      });
  });
```

```
  // Use separate effect for Results of sort so it is not ovverriden by
updates to Products.
  useEffect(() => {
    if (initialized === false) {
      fetch("http://localhost:8080/shop")
        .then((response) => response.json())
        .then((results) => {
          setResults(results);
          setInitialized(true);
        });
    }
  });

  useEffect(() => {
    handleSearch();
  }, [Animal, Type]);

  function addToCart(product) {
    if (product.stock != 0) {
      let copy = [...Cart, product];
      setCart(copy);
    } else {
      alert(
        `Product: ${product.title}\n` +
          `is currently out of stock. We are sorry for the inconvenience!`
      );
    }
  }

  function removeFromCart(product) {
    let cartCopy = [];
    let removeIndex = -1;
    for (let i = 0; i < Cart.length; i++) {
      if (Cart[i].id === product.id) {
        removeIndex = i;
        break;
      }
    }

    let copyInd = 0;
```

```
  for (let i = 0; i < Cart.length; i++) {
    if (i != removeIndex) {
      cartCopy[copyInd] = Cart[i];
      copyInd++;
    }
  }
  setCart(cartCopy);
}

function handleSearch() {
  var value = document.getElementById("search").value;
  const results = Products.filter((p) => {
    if (Animal === "all" && Type === "all") {
      return p.title.toLowerCase().includes(value.toLowerCase());
    } else if (Animal === "all" && Type !== "all") {
      return (
        p.title.toLowerCase().includes(value.toLowerCase()) &&
        p.type.toLowerCase() === Type
      );
    } else if (Animal !== "all" && Type === "all") {
      return (
        p.title.toLowerCase().includes(value.toLowerCase()) &&
        p.animal.toLowerCase() === Animal
      );
    } else {
      return (
        p.title.toLowerCase().includes(value.toLowerCase()) &&
        p.animal.toLowerCase() === Animal &&
        p.type.toLowerCase() === Type
      );
    }
  });
  setResults(results);
}

function quantity(productId) {
  let amount = Cart.filter((item) => item.id === productId);
  return amount.length;
}
```

```
const getStars = (product) => {
  const rating = product.rate.rating;
  const count = product.rate.count;
  if (rating >= 4.5) {
    return (
      <div>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/> ({count})
      </div>
    );
  } else if (rating >= 3.5) {
    return (
      <div>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/> ({count})
      </div>
    );
  } else if (rating >= 2.5) {
    return (
      <div>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/> ({count})
      </div>
    );
  } else if (rating >= 1.5) {
    return (
      <div>
        <img src="images/star.svg" alt="star"/>
        <img src="images/star.svg" alt="star"/> ({count})
      </div>
    );
  } else if (rating >= 0.5) {
    return (
      <div>
```

```
          <img src="images/star.svg" alt="star"/> ({count})
        </div>
      );
    } else {
      return <div>Zero Stars ({count})</div>;
    }
  };

  return (
    <div>
      <div
        id="main"
        className="p-4 p-md-5 mb-4 rounded text-body-emphasis
bg-body-secondary"
      >
        <h1 className="display-4">Browse Our Pet Supplies</h1>
        <p className="lead my-3">
          Looking for some treats for your pet? Or perhaps some new toys
for your
          recent adoption? We have you covered! Take a look at our wide
          selection down below!
        </p>
      </div>

      <div
        style={{ textAlign: "center", fontSize: 20 + "px" }}
        className="row g-3"
      >
        <div
          style={{ width: 250 + "px", marginLeft: 50 + "px" }}
          className="col-md-5"
        >
          <label htmlFor="search" className="form-label">
            Search
          </label>
          <input
            type="text"
            className="form-control"
            id="search"
            placeholder="Search"
```

```
            onChange={handleSearch}
          />
        </div>

        <div
          style={{ width: 250 + "px", marginLeft: 25 + "px" }}
          className="col-md-5"
        >
          <label htmlFor="animal" className="form-label">
            Sort by Animals
          </label>
          <select
            onChange={(e) => {
              setAnimal(e.target.value);
            }}
            className="form-select"
            id="animal"
          >
            <option value="all">All</option>
            <option value="cat">For Cats</option>
            <option value="dog">For Dogs</option>
          </select>
          <div className="invalid-feedback">Please make a valid
selection.</div>
        </div>

        <div
          style={{
            width: 250 + "px",
            marginLeft: 25 + "px",
            paddingBottom: 25 + "px",
          }}
          className="col-md-5"
        >
          <label htmlFor="product" className="form-label">
            Sort by Product
          </label>
          <select
            onChange={(e) => {
              setType(e.target.value);
```

```
            }}
            className="form-select"
            id="type"
          >
            <option value="all">All</option>
            <option value="food">Food and Treats</option>
            <option value="toy">Toys</option>
          </select>
          <div className="invalid-feedback">Please make a valid
selection.</div>
        </div>
        <hr style={{ marginTop: 25 + "px", width: 90 + "%", margin: "auto"
}} />

        <div style={{ backgroundColor: "#d9d9d9", paddingTop: 40 + "px"
}}>
          <div className="container">
            <div
              className="row row-cols-1 row-cols-sm-2 row-cols-md-3
row-cols-lg-4 g-3"
              style={{ paddingBottom: 20 + "px" }}
            >
              {Results.map((product) => (
                <div key={product.id} className="col">
                  <div className="card shadow-sm">
                    <img
                      src={product.image}
                      className="card-img-top"
                      alt="Item Image"
                    ></img>
                    <div className="card-body">
                      <p className="card-text" style={{ fontSize: 15 +
"px" }}>
                        {" "}
                        <strong style={{ fontSize: 20 + "px" }}>
                          {product.title}
                        </strong>
                        <br />({product.animal}/{product.type})
                      </p>
                      <p className="card-text">{product.price}</p>
```

```
                    {getStars(product)}
                    <hr />
                    <div>
                      <button onClick={() => removeFromCart(product)}>
                        <img
                          src="images/remove.svg"
                          width="100%"
                          height="auto"
                          alt="remove"
                        />
                      </button>
                      <span style={{ margin: 15 + "px" }}>
                        {quantity(product.id)}
                      </span>
                      <button onClick={() => addToCart(product)}>
                        <img
                          src="images/add.svg"
                          width="100%"
                          height="auto"
                          alt="add"
                        />
                      </button>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          </div>
        </div>
      </div>
    );
}

export default Shop;
```

src/Checkout.js

```
import React, { useContext } from "react";
import { useState, useEffect } from "react";
import { Context } from "./Context";
import Button from "react-bootstrap/Button";
import Container from "react-bootstrap/Container";
import Form from "react-bootstrap/Form";
import {
  List,
  ListItem,
  Card,
  Typography,
} from "@material-tailwind/react";
import { Rating } from "react-simple-star-rating";

function Checkout() {
  const [View, setView] = useState(0);
  const { cart, order } = useContext(Context);
  const [Cart, setCart] = cart;
  const [Order, setOrder] = order;
  const [SelectedItems, setSelectedItems] = useState([]);

  useEffect(() => {
    console.log(Cart);
    getSelectedItems(Cart);
  }, [Cart]);

  useEffect(() => {
    console.log(SelectedItems);
  }, [SelectedItems]);

  function quantity(items, productId) {
    let amount = items.filter((item) => item.id === productId);
    return amount.length;
  }

  //Get each item in cart, but do not duplicated repeats. Only display
each item once.
  function getSelectedItems() {
    const items = [];
```

```
  const item = {};

  for (let i in Cart) {
    const product = Cart[i]["id"];
    item[product] = Cart[i];
  }

  for (let i in item) {
    items.push(item[i]);
  }

  setSelectedItems(items);
}

const getPrice = (cart) => {
  let subtotal = 0;
  for (let i = 0; i < cart.length; i++) {
    if (cart[i].name == null) {
      let price = Number(String(cart[i].price).substring(1));
      subtotal += price;
    }
  }

  const tax = Number(subtotal * 0.06).toFixed(2);
  let total = Number(subtotal) + Number(tax);
  total = total.toFixed(2);

  if (cart.length > 0) {
    return (
      <div style={{ paddingTop: 30 + "px" }}>
        <h3>Subtotal: ${subtotal}</h3>
        <h3>Tax: ${tax}</h3>
        <h3>Total: ${total}</h3>
      </div>
    );
  }
};

function displayCart(items, entity) {
  if (entity.title != null) {
```

```
    return (
      <div>
        <p className="card-text">{entity.price} each</p>
        <span style={{ margin: 15 + "px" }}>
          Quantity: {quantity(items, entity.id)}
        </span>
      </div>
    );
  } else {
    return <span>Place on hold</span>;
  }
}

const Confirm = () => {
  const [Validated, setValidated] = useState(false);
  const [FormData, setFormData] = useState({
    name: "",
    email: "",
    address: "",
    city: "",
    state: "",
    zip: "",
    card: "",
  });

  function handleSubmit(event) {
    event.preventDefault();
    const form = event.target;
    if (form.checkValidity() === false) {
      event.preventDefault();
      event.stopPropagation();
      setValidated(true);
    } else {
      confirmOrder(event);
    }
  }

  function confirmOrder(event) {
    const form = event.target;
    const name = form.name.value;
```

```
    const email = form.email.value;
    const address = form.address.value;
    const city = form.city.value;
    const state = form.state.value;
    const zip = form.zip.value;
    const card = form.card.value;

    const json = {
      name: name,
      email: email,
      address: address,
      city: city,
      state: state,
      zip: zip,
      card: card,
      items: Cart,
    };

    setOrder(json);

    // Post order
    fetch("http://localhost:8080/order", {
      method: "POST",
      body: JSON.stringify(json),
      headers: {
        "Content-type": "application/json; charset=UTF-8",
      },
    });

    // Update product stock (ignore animal in cart)
    for (let i in SelectedItems) {
      if (SelectedItems[i].title != null) {
        const item = SelectedItems[i];
        const stock = Number(item.stock);
        const newStock = stock - quantity(Cart, item.id);

        fetch("http://localhost:8080/shop/" + item.id, {
          method: "PUT",
          headers: { "content-type": "application/json" },
          body: JSON.stringify({
```

```
            id: item.id,
            type: item.type,
            animal: item.animal,
            title: item.title,
            price: item.price,
            image: item.image,
            stock: String(newStock),
            rate: item.rate,
          }),
        });
      }
    }

    // Update product stock (ignore animal in cart)
    for (let i in SelectedItems) {
      if (SelectedItems[i].title == null) {
        const item = SelectedItems[i];
        console.log(item);

        fetch("http://localhost:8080/adopt/" + item.id, {
          method: "PUT",
          headers: { "content-type": "application/json" },
          body: JSON.stringify({
            id: item.id,
            animal: item.animal,
            name: item.name,
            age: item.age,
            description: item.description,
            image: item.image,
            isReserved: true,
          }),
        });
      }
    }

    setCart([]);

    form.name.value = "";
    form.email.value = "";
    form.address.value = "";
```

```
      form.city.value = "";
      form.state.value = "";
      form.zip.value = "";
      form.card.value = "";

      //alert(`Your order has been placed. Thanks for your support!`);
      setView(1);
    }

    const change = (event) => {
      const { name, value } = event.target;
      setFormData({
        ...FormData,
        [name]: value,
      });
    };

    function getCheckout() {
      // Only display checkout form if cart is not empty.
      if (Cart.length > 0) {
        return (
          <div>
            <hr style={{ width: 90 + "%", margin: "auto" }} />
            <Container style={{ marginTop: 15 + "px" }}>
              <h1 style={{ textAlign: "center" }} className="font">
                Checkout
              </h1>
              <Form noValidate validated={Validated}
onSubmit={handleSubmit}>
                <Form.Group className="mb-3" controlId="name">
                  <Form.Label>Name</Form.Label>
                  <Form.Control
                    type="text"
                    name="name"
                    placeholder="Name"
                    value={FormData.name}
                    onChange={change}
                    pattern="^[a-zA-Z ]+$"
                    minLength={2}
                    required
```

```
                    isInvalid={Validated && !/^[a-zA-Z
]+$/.test(FormData.name)}
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter a name (letters only)
                  </Form.Control.Feedback>
                </Form.Group>
                <Form.Group className="mb-3" controlId="email">
                  <Form.Label>Email</Form.Label>
                  <Form.Control
                    type="text"
                    name="email"
                    value={FormData.email}
                    onChange={change}
                    placeholder="123@gmail.com"
                    pattern="^[^\s@]+@[^\s@]+\.[^\s@]+$"
                    required
                    isInvalid={
                      Validated &&
                      !/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(FormData.email)
                    }
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter a valid email address
                  </Form.Control.Feedback>
                </Form.Group>
                <Form.Group className="mb-3" controlId="address">
                  <Form.Label>Address</Form.Label>
                  <Form.Control
                    type="text"
                    placeholder="1234 Happy Rd"
                    required
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter an address
                  </Form.Control.Feedback>
                </Form.Group>
                <Form.Group className="mb-3" controlId="city">
                  <Form.Label>City</Form.Label>
                  <Form.Control
```

```
                    type="text"
                    name="city"
                    placeholder="City"
                    value={FormData.city}
                    onChange={change}
                    pattern="^[a-zA-Z]+$"
                    minLength={2}
                    required
                    isInvalid={Validated &&
!/^[a-zA-Z]+$/.test(FormData.city)}
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter a city (letters only)
                  </Form.Control.Feedback>
                </Form.Group>
                <Form.Group className="mb-3" controlId="state">
                  <Form.Label>State</Form.Label>
                  <Form.Control
                    type="text"
                    name="state"
                    placeholder="State"
                    value={FormData.state}
                    onChange={change}
                    pattern="^[a-zA-Z]+$"
                    required
                    isInvalid={Validated &&
!/^[a-zA-Z]+$/.test(FormData.state)}
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter a state name (letters only)
                  </Form.Control.Feedback>
                </Form.Group>
                <Form.Group className="mb-3" controlId="zip">
                  <Form.Label>Zip Code</Form.Label>
                  <Form.Control
                    type="text"
                    name="zip"
                    placeholder="12345"
                    value={FormData.zip}
                    onChange={change}
```

```
                    pattern="[0-9]{5}"
                    required
                    isInvalid={Validated &&
!/^[0-9]{5}$/.test(FormData.zip)}
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter a five digit zip code (numerical inputs
only)
                  </Form.Control.Feedback>
                </Form.Group>
                <Form.Group className="mb-3" controlId="card">
                  <Form.Label>Card Number</Form.Label>
                  <Form.Control
                    type="text"
                    name="card"
                    placeholder="**** **** **** ****"
                    value={FormData.card}
                    onChange={change}
                    pattern="[0-9]{16}"
                    required
                    isInvalid={Validated &&
!/^[0-9]{16}$/.test(FormData.card)}
                  />
                  <Form.Control.Feedback type="invalid">
                    Please enter a valid card number (16 numerical digit
only)
                  </Form.Control.Feedback>
                </Form.Group>
                <Button type="submit">Place Order</Button>
              </Form>
            </Container>
          </div>
        );
      } else {
        return (
          <p style={{ textAlign: "center", fontSize: 20 + "px" }}>
            There are currently no items in your cart.
          </p>
        );
      }
```

```
    }

    return (
      <div>
        <div
          style={{
            paddingTop: 10 + "px",
            paddingBottom: 15 + "px",
            textAlign: "center",
            fontSize: 20 + "px",
          }}
        >
          <h1 className="font">Your Cart</h1>
          <hr style={{ width: 90 + "%", margin: "auto" }} />
          <div
            className="container"
            style={{ marginTop: 25 + "px", marginBottom: 10 + "px" }}
          >
            <div
              id="col"
              className="row row-cols-1 row-cols-sm-2 row-cols-md-3
row-cols-lg-4 g-3"
            >
              {SelectedItems.map((product) => (
                <div key={product.id} className="col">
                  <div className="card shadow-sm">
                    <img
                      src={product.image}
                      className="card-img-top"
                      alt="Item Image"
                    ></img>
                    <div className="card-body">
                      <p className="card-text" style={{ fontSize: 15 +
"px" }}>
                        {" "}
                        <strong style={{ fontSize: 20 + "px" }}>
                          {product.title}
                          {product.name}
                        </strong>
                        <br />
```

```
                    </p>
                    <div>{displayCart(Cart, product)}</div>
                  </div>
                </div>
              </div>
            ))}
          </div>
        </div>
        <div>{getPrice(Cart)}</div>
      </div>
      {getCheckout()}
    </div>
  );
};

const Review = () => {
  const [ratings, setRatings] = useState({});
  const [submitted, setSubmitted] = useState([]);
  const [purchases, setPurchases] = useState([]); // Local copy of
purchased items so cart can be reset prior
  const [items, setItems] = useState([]); // Local copy of purchased
items without duplicates

  useEffect(() => {
    setPurchases(Order.items);
  }, [Order]);

  useEffect(() => {
    getItems();
  }, [purchases]);

  useEffect(() => {
    window.scrollTo(0, 0);
  }, []);

  const handleRating = (productId, rating) => {
    setRatings((prevRatings) => ({
      ...prevRatings,
      [productId]: rating,
    }));
```

```
};

function getItems() {
  const items = [];
  const item = {};

  for (let i in purchases) {
    const product = purchases[i]["id"];
    item[product] = purchases[i];
  }

  for (let i in item) {
    items.push(item[i]);
  }

  setItems(items);
}

const submit = (product) => {
  if (!submitted.includes(product.id)) {
    let inputRating = ratings[product.id];
    if (inputRating !== undefined) {
      inputRating += 1;
      const rating = Number(product.rate.rating);
      let count = Number(product.rate.count);
      let newRating = Number(
        (rating * count + inputRating) / ++count
      ).toFixed(2);

      fetch("http://localhost:8080/shop/" + product.id, {
        method: "PUT",
        headers: { "content-type": "application/json" },
        body: JSON.stringify({
          id: product.id,
          type: product.type,
          animal: product.animal,
          title: product.title,
          price: product.price,
          image: product.image,
          stock: product.stock,
```

```
            rate: {
              rating: String(newRating),
              count: String(count),
            },
          }),
        });
        setSubmitted([...submitted, product.id]);
      }
    } else {
      alert(`You have already left a review for this product.`);
    }
  };

  const filterOutAnimal = () => {
    let copy = [];
    for (let i = 0; i < items.length; i++) {
      if (items[i].title != null) {
        copy.push(items[i]);
      }
    }
    return copy;
  };

  function getCard() {
    let card = String(Order.card).substring(12);
    return "************" + card;
  }

  return (
    <div
      style={{
        paddingTop: 10 + "px",
        paddingBottom: 15 + "px",
        textAlign: "center",
        fontSize: 20 + "px",
      }}
    >
      <div
        id="main"
```

```
          className="p-4 p-md-5 mb-4 rounded text-body-emphasis
bg-body-secondary"
        >
          <h1 className="display-4">Thank You For Your Purchase!</h1>
          <p className="lead my-3">
            Your support goes a long way in supporting our small business.
We
            strive to provide excellence in the products we sell. If you
            experience any difficulties with your purchase, please do not
            hesitate to contact us!
          </p>
        </div>
        <h1 className="font">Your Order Details</h1>
        <div
          className="container"
          style={{ marginTop: 25 + "px", marginBottom: 20 + "px" }}
        >
          <div
            id="col"
            className="row row-cols-1 row-cols-sm-2 row-cols-md-3
row-cols-lg-4 g-3"
          >
            {items.map((product) => (
              <div key={product.id} className="col">
                <div className="card shadow-sm">
                  <img
                    src={product.image}
                    className="card-img-top"
                    alt="Item Image"
                  ></img>
                  <div className="card-body">
                    <p className="card-text" style={{ fontSize: 15 + "px"
}}>
                      {" "}
                      <strong style={{ fontSize: 20 + "px" }}>
                        {product.title}
                        {product.name}
                      </strong>
                      <br />
                    </p>
```

```
                    <div>{displayCart(purchases, product)}</div>
                  </div>
                </div>
              </div>
            ))}
          </div>
          <div>
            <div id="authors">
              <div style={{ width: 100 + "%", float: "inline-end" }}>
                <h3
                  style={{ marginTop: 20 + "px", marginBottom: 20 + "px"
}}
                  className="font"
                >
                  Billing Information
                </h3>
                <div
                  className="author"
                  style={{ float: "left", marginLeft: 2.5 + "%" }}
                >
                  <div>
                    <h4>Shipping Address</h4>
                    {Order.address}
                    <br /> {Order.city} {", "} {Order.state} {Order.zip}
                  </div>
                </div>

                <div
                  className="author"
                  style={{ float: "right", marginRight: 2.5 + "%" }}
                >
                  <div>
                    <h4>Payment</h4>
                    Name: {Order.name} <br />
                    Card Number: {getCard()}
                  </div>
                </div>
              </div>
            </div>
          </div>
```

```
          {getPrice(purchases)}
        </div>
        <hr
          style={{ width: 90 + "%", margin: "auto", marginBottom: 15 +
"px" }}
        />
        <div>
          <h1 className="display-6">
            Would you Like to Leave Any Product Ratings?
          </h1>
          <p>
            Leaving a rating for any of our products helps us greatly
understand
            the needs of our customers. It helps us decide which items to
keep
            in our shop. So, any feedback provided would be greatly
appreciated!
          </p>
        </div>
        <Card className="w-96">
          <List>
            {filterOutAnimal().map((product) => (
              <ListItem key={product.id}>
                <div>
                  <Typography variant="h6" color="blue-gray">
                    {product.title}
                  </Typography>
                  <div className="App">
                    <Rating
                      onClick={(newValue) => handleRating(product.id,
newValue)}
                      initialValue={0}
                    />
                    <Button
                      style={{ margin: 5 + "px" }}
                      onClick={() => submit(product)}
                    >
                      Submit
                    </Button>
                  </div>
```

```
                </div>
              </ListItem>
            ))}
          </List>
        </Card>
        <Button
          onClick={() => {
            setView(0);
          }}
        >
          Back
        </Button>
      </div>
    );
  };

  return (
    <div>
      {View === 0 && <Confirm />}
      {View === 1 && <Review />}
    </div>
  );
}

export default Checkout;
```

src/Context.js

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

export const Context = React.createContext();
export const ContextProvider = ({ children }) => {
  const [cart, setCart] = useState([]);
  const [order, setOrder] = useState({});

  return (
    <Context.Provider value={{ cart: [cart, setCart], order: [order,
setOrder] }}>{children}</Context.Provider>
  );
```

```
};
```

src/Authors.js

```
function Authors() {
  return (
    <div>
      <div
        id="main"
        className="p-4 p-md-5 mb-4 rounded text-body-emphasis
bg-body-secondary"
      >
        <h1 className="display-4">SE/ComS 319: Construction of User
Interfaces</h1>
        <h2 className="display-6">Spring 2024, Dr. Abraham Aldaco</h2>
      </div>

      <div id="authors">
        <div style={{width:100 + "%", float: "inline-end"}}>
          <h3 style={{marginTop: 20 + "px", marginBottom: 20 + "px"}}
className="display-6">
            Authors: Team77
          </h3>
          <div className="author" style={{float:"left", marginLeft: 2.5 +
"%"}}>
            <h2>Justin Sebahar</h2>
            <h3>jtseb@iastate.edu</h3>
            <h4>5/3/2024</h4>
          </div>

          <div className="author" style={{float:"right", marginRight: 2.5
+ "%"}}>
            <h2>Carter Peterson</h2>
            <h3>cjpetey@iastate.edu</h3>
            <h4>5/3/2024</h4>
          </div>
        </div>
```

```
        <div style={{width:100 + "%", float: "inline-end", marginTop: 20 +
"px"}}>
          <h4>
            Meet Scooter the turtle and Chloe the dog; they're the
inspiration
            behind this project!
          </h4>
          <img
            src="./images/author-scooter.jpg"
            style={{marginBottom:25 + "px", marginRight: 5 + "px"}}
            width="30%"
            height="auto"
            alt="Scooter the Turtle"
          />
          <img
            src="./images/author-chloe.jpg"
            style={{marginBottom:25 + "px", marginLeft: 5 + "px"}}
            width="30%"
            height="auto"
            alt="Chloe the Dog"
          />
        </div>
      </div>
    </div>
  );
}

export default Authors;
```

src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import "bootstrap/dist/css/bootstrap.css";
import { ContextProvider } from "./Context";

const root = ReactDOM.createRoot(document.getElementById("root"));
```

```
root.render(
  <ContextProvider>
    <App />
  </ContextProvider>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

arc/App.css

```
header {
  background-color: white;
  color: black;
  font-family: 'Comic Sans MS', cursive;
}

header span {
  font-size: 20px;
  font-style: bold;
  color: black
}

#main {
  text-align: center;
  padding-top: 50px;
}

#footer{
  background-color:rgb(233 236 239);
  margin-left:0px;
  margin-right:0px;
  margin-top:75px;
  height:75px;
  width:100%;
}
```

```
#services{
  text-align: center;
  padding-top:10px;
  margin-bottom:25px;
  font-family: 'Comic Sans MS', cursive;
}

.font{
  font-family: 'Comic Sans MS', cursive;
}

.hover:hover{
  text-decoration: underline;
}

.service{
  width:45%;
  margin-bottom:25px;
  border:solid 2px;
  background-color: rgb(233 236 239);
}

#services a{
  text-decoration: underline;
  color: black;
}

#authors{
  text-align: center;
  padding-top:10px;
  margin-bottom:25px;
}

.author{
  width:45%;
  margin-bottom:25px;
  border:solid 2px;
  background-color: rgb(233 236 239);
}
```

package.json

```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@material-tailwind/react": "^2.1.9",
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "bootstrap": "^5.3.3",
    "feedback": "^0.3.2",
    "list-item": "^2.0.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.10.2",
    "react-dom": "^18.2.0",
    "react-listview": "^0.1.0",
    "react-scripts": "5.0.1",
    "react-simple-star-rating": "^5.1.7",
    "tailwind": "^4.0.0",
    "tailwindcss": "^3.4.3",
    "use-between": "^1.3.5",
    "use-context": "^0.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
```

```
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Backend

index.js

```
var express = require("express");
var cors = require("cors");
var app = express();
var fs = require("fs");
var bodyParser = require("body-parser");
app.use(cors());
app.use(bodyParser.json());

const { MongoClient } = require("mongodb");

const url = "mongodb://127.0.0.1:27017";
const dbName = "SE319FINAL";
const client = new MongoClient(url);
const db = client.db(dbName);

const port = "8080";
const host = "localhost";

app.get("/shop", async (req, res) => {
  try {
    await client.connect();
    const query = {};
    const results = await db.collection("shop").find(query).toArray();
```

```
    console.log(results);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

app.get("/shop/:id", async (req, res) => {
  try {
    const product = Number(req.params.id);
    const query = { id: String(product) };
    await client.connect();

    // Check if the 'product' exists
    const check = await db.collection("shop").find(query).toArray();
    if (check.length == 0) {
      const msg = "Bad request: Item does not exist";
      console.log(msg);
      return res.status(409).send({ error: msg });
    }

    const results = await db.collection("shop").find(query).toArray();
    console.log(results);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

app.post("/shop", async (req, res) => {
  try {
    await client.connect();

    // Check if body is non empty
    if (!req.body || Object.keys(req.body).length === 0) {
      const msg = "Bad request: No data provided.";
      console.log(msg);
      return res.status(400).send({ error: msg });
```

```
    }

    // Check if the 'product' exists
    const itemId = req.body.id;
    const query = { id: itemId };
    const results = await db.collection("shop").find(query).toArray();
    if (results.length > 0) {
      const msg = "Bad request: Item already exists";
      console.log(msg);
      return res.status(409).send({ error: msg });
    }

    const newDocument = {
      id: req.body.id,
      type: req.body.type,
      animal: req.body.animal,
      title: req.body.title,
      price: req.body.price,
      image: req.body.image,
      rate: {
        rating: req.body.rate.rating,
        count: req.body.rate.count,
      },
    };
    console.log(newDocument);

    const response = await db.collection("shop").insertOne(newDocument);
    res.status(200);
    res.send(response);
  } catch (error) {
    res.status(500).send({ error: "An internal server error occurred" });
  }
});

app.put("/shop/:id", async (req, res) => {
  try {
    const id = String(req.params.id);
    const query = { id: id };
    await client.connect();
```

```
    // Check if body is non empty
    if (!req.body || Object.keys(req.body).length === 0) {
      const msg = "Bad request: No data provided.";
      console.log(msg);
      return res.status(400).send({ error: msg });
    }

    // Check if the 'product' exists
    const check = await db.collection("shop").find(query).toArray();
    if (check.length == 0) {
      const msg = "Bad request: Item does not exist";
      console.log(msg);
      return res.status(409).send({ error: msg });
    }

    console.log(req.body);
    const updateData = {
      $set: {
        id: req.body.id,
        type: req.body.type,
        animal: req.body.animal,
        title: req.body.title,
        price: req.body.price,
        image: req.body.image,
        stock: req.body.stock,
        rate: {
          rating: req.body.rate.rating,
          count: req.body.rate.count,
        },
      },
    };

    const options = { upsert: true };
    const results = await db
      .collection("shop")
      .updateOne(query, updateData, options);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
```

```
  }
});

app.delete("/shop/:id", async (req, res) => {
  try {
    const id = String(req.params.id);
    const query = { id: id };
    await client.connect();

    // Check if the 'product' exists
    const check = await db.collection("shop").find(query).toArray();
    if (check.length == 0) {
      const msg = "Bad request: Item does not exist";
      console.log(msg);
      return res.status(409).send({ error: msg });
    }

    const results = await db.collection("shop").deleteOne(query);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});
//-----------------------------------

app.get("/adopt", async (req, res) => {
  try {
    await client.connect();
    const query = {};
    const results = await db.collection("adopt").find(query).toArray();
    console.log(results);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

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

```
  try {
    const id = Number(req.params.id);
    const query = { id: id };
    await client.connect();

    // Check if the 'product' exists
    const check = await db.collection("adopt").find(query).toArray();
    if (check.length == 0) {
      const msg = "Bad request: Item does not exist";
      console.log(msg);
      return res.status(409).send({ error: msg });
    }

    const results = await db.collection("adopt").find(query).toArray();
    console.log(results);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

app.post("/adopt", async (req, res) => {
  try {
    await client.connect();

    // Check if body is non empty
    if (!req.body || Object.keys(req.body).length === 0) {
      const msg = "Bad request: No data provided.";
      console.log(msg);
      return res.status(400).send({ error: msg });
    }

    // Check if the 'product' exists
    const id = req.body.id;
    const query = { id: id };
    const results = await db.collection("adopt").find(query).toArray();
    if (results.length > 0) {
      const msg = "Bad request: Item already exists";
      console.log(msg);
```

```
      return res.status(409).send({ error: msg });
    }

    const newDocument = {
      id: req.body.id,
      animal: req.body.animal,
      name: req.body.name,
      age: req.body.age,
      description: req.body.description,
      image: req.body.image,
      isReserved: req.body.isReserved,
    };
    console.log(newDocument);

    const response = await db.collection("adopt").insertOne(newDocument);
    res.status(200);
    res.send(response);
  } catch (error) {
    res.status(500).send({ error: "An internal server error occurred" });
  }
});

app.put("/adopt/:id", async (req, res) => {
  try {
    const id = req.params.id;
    const query = { id: id };
    await client.connect();

    // Check if body is non empty
    if (!req.body || Object.keys(req.body).length === 0) {
      const msg = "Bad request: No data provided.";
      console.log(msg);
      return res.status(400).send({ error: msg });
    }

    // Check if the 'product' exists
    const check = await db.collection("adopt").find(query).toArray();
    if (check.length == 0) {
      const msg = "Bad request: Item does not exist";
      console.log(msg);
```

```
      return res.status(409).send({ error: msg });
    }

    console.log(req.body);
    const updateData = {
      $set: {
      id: req.body.id,
      animal: req.body.animal,
      name: req.body.name,
      age: req.body.age,
      description: req.body.description,
      image: req.body.image,
      isReserved: req.body.isReserved,
      },
    };

    const options = { upsert: true };
    const results = await db
      .collection("adopt")
      .updateOne(query, updateData, options);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

app.delete("/adopt/:id", async (req, res) => {
  try {
    const id = req.params.id;
    const query = { id: id };
    await client.connect();

    // Check if the 'product' exists
    const check = await db.collection("adopt").find(query).toArray();
    if (check.length == 0) {
      const msg = "Bad request: Item does not exist";
      console.log(msg);
      return res.status(409).send({ error: msg });
    }
```

```
    const results = await db.collection("adopt").deleteOne(query);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

app.get("/order", async (req, res) => {
  try {
    await client.connect();
    const query = {};
    const results = await db.collection("order").find(query).toArray();
    console.log(results);
    res.status(200);
    res.send(results);
  } catch (error) {
    res.status(500).send({ message: "Internal Server Error" });
  }
});

app.post("/order", async (req, res) => {
  try {
    await client.connect();

    // Check if body is non empty
    if (!req.body || Object.keys(req.body).length === 0) {
      const msg = "Bad request: No data provided.";
      console.log(msg);
      return res.status(400).send({ error: msg });
    }

    const newDocument = {
      name: req.body.name,
      email: req.body.email,
      address: req.body.address,
      city: req.body.city,
      state: req.body.state,
      zip: req.body.zip,
```

```
      card: req.body.card,
      items: req.body.items,
    };
    console.log(newDocument);

    const response = await db.collection("order").insertOne(newDocument);
    res.status(200);
    res.send(response);
  } catch (error) {
    res.status(500).send({ error: "An internal server error occurred" });
  }
});

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

package.json

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "team77",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.19.2",
    "mongodb": "^6.5.0",
    "node": "^18.20.2",
    "nodemon": "^3.1.0"
  }
}
```