Faculty of Engineering and Architecture

Computer Engineering

Web Programming

Final Project

2020-2021

Spring Semester

Ceyhun Bağrıaçık

170303003

Dr. Pınar Karadayı Ataş

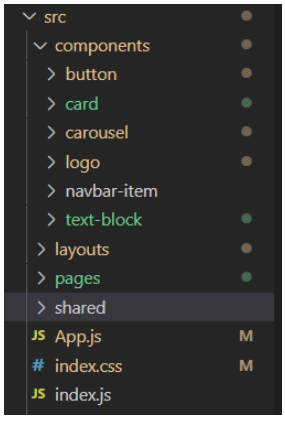
Project definition: In this project, I wanted to create a website that sells cards from the famous TV Anime "Yugioh". As for the tools, I have used React JS to code the frontend to make it look responsive and taken help from my browser's developer tools while designing the frontend.

Packages Used in the Project: I used yarn as my package manager and utilised following packages in my code:

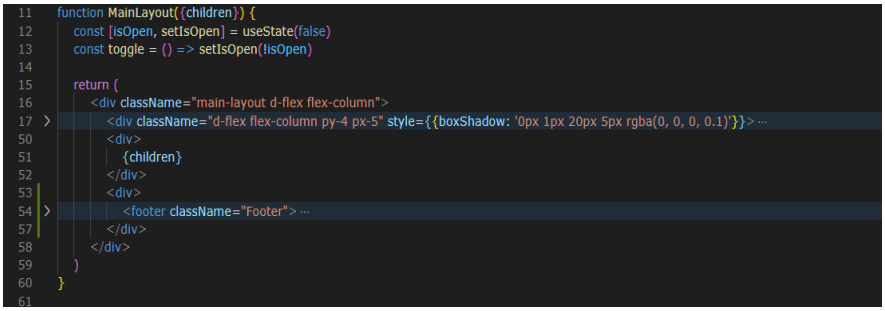
- fontawesome - for icons
- bootstrap - for premade css classes
- reactstrap - for premade components that is made specifically made for react
- react - to be able to use UseState to change states of the variables
- react-dom - to render the page.
- react-router-dom - for router component for easy pathing from page to page.

Component system in ReactJS: Separating each item as a component and working solely on an item to use that item whenever you want is one of the greatest strengths of React therefore I wanted to utilise that strength to create my website.

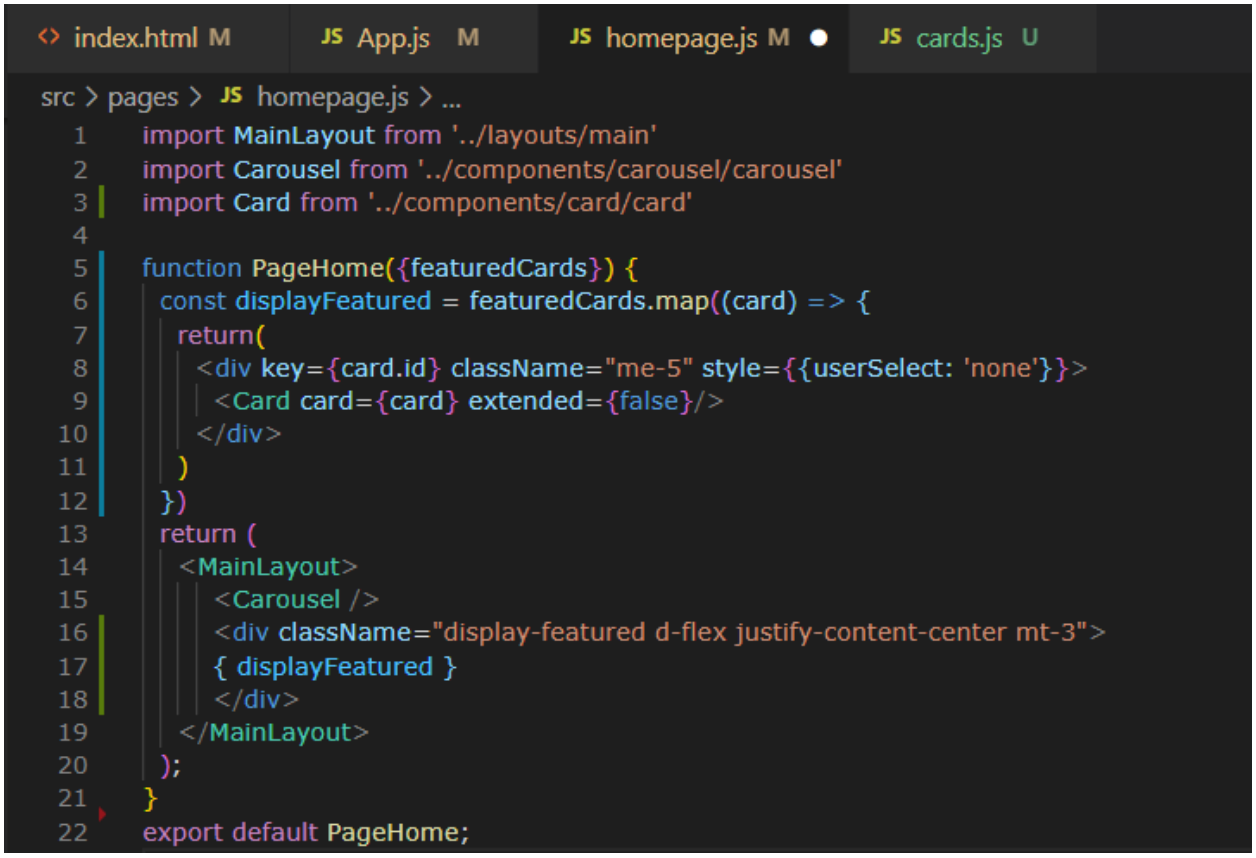
On the right side is the structure of my website. At first I created a shared folder that contains the color palette of the website. Another important section is the layouts part. There exists main.js and main.css files. In main.js There is the main layout of a page's structure. Navbar at the top with the logo of the website and footer at the bottom that shows the copyright. That main layout component takes a prop which fills out the body of the page.



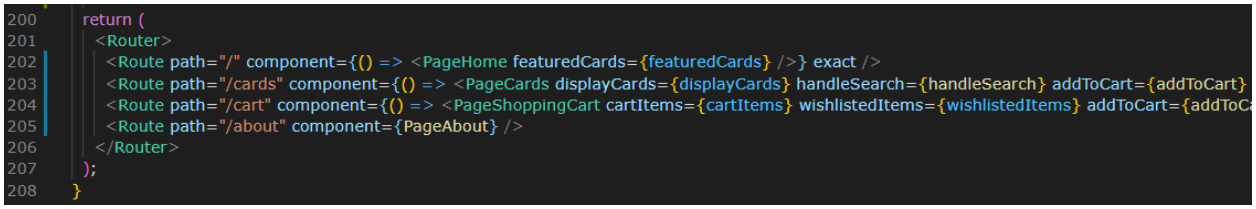
Children prop that has been passed, fills in between the navbar and footer.
In my case I separated each page



to have it’s own file.

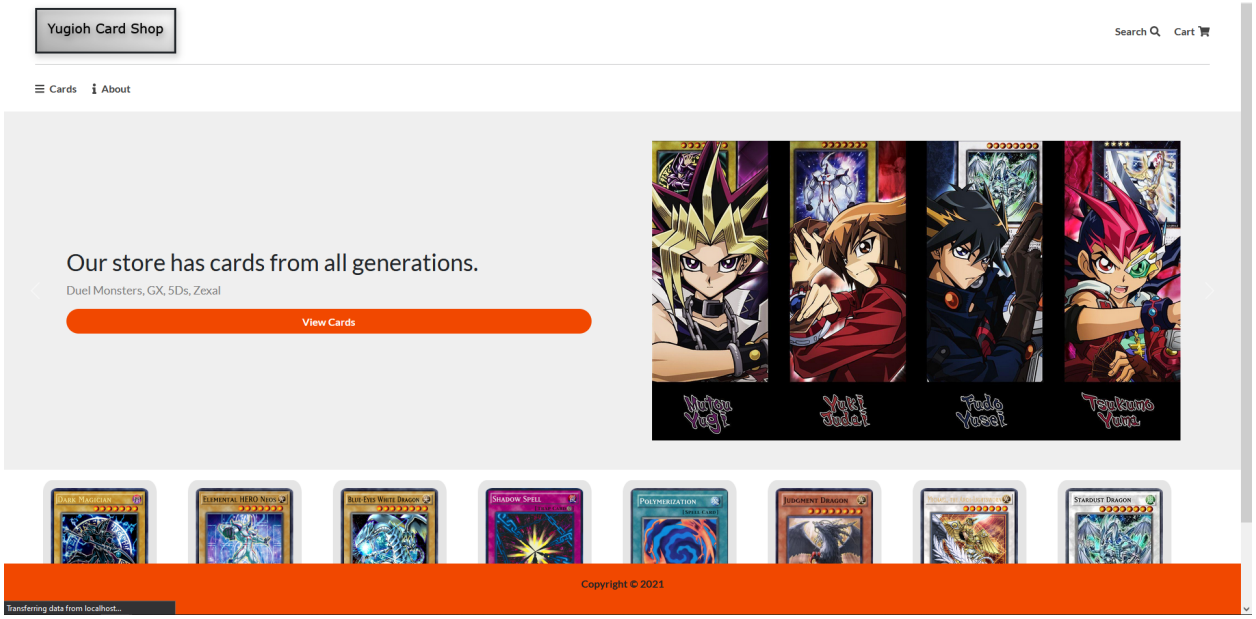


Above is the homepage.js it returns <MainLayout>Content for the homepage</MainLayout> In my case a carousel that has an interactable button and below displaying featured cards. displayFeatured function runs for each card inside the featuredCards list that has been passed down with props.



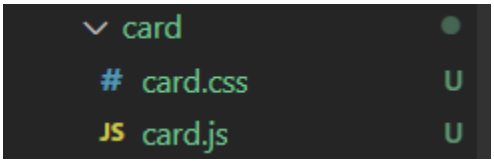
Above is the routing section of my code in App.js, Main functions that affect the variables and some of the required variables for displaying some elements are passed through props to subcomponents. For example featuredCards are passed via props to be displayed later on.

Visual of my homepage:



Here navbar-items act as a link that forwards the user to other pages, as well as the button on the carousel.

Each component has it's .js and .css files. Here I'll show the card.js file.



src > components > card > JS card.js > ...

```
1  import './card.css'
2  import Button from '../button/button'
3  import { useState } from 'react'
4  import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
5  import { faHeart as farHeart } from '@fortawesome/free-regular-svg-icons'
6  import { faHeart as fasHeart } from '@fortawesome/free-solid-svg-icons'
7
8  const Card = ({card, extended, onButtonClickAction, onIconClickAction, inCart, inWishlist}) => {
9      const  [isHovered, setIsHovered] = useState(false);
10     return (
11         <div className="whole-wrap d-flex flex-column">
12             <div className="container card-wrap ">
13                 <img src={card.cardImage} alt=""/>
14             </div>
15             {
16                 extended &&
17                 <div className="container card-info">
18                     <p>{card.name.substring(0,23)}</p>
19                     <h6>{card.edition}</h6>
20                     <h4>{card.price}$</h4>
21                     <div className="clickable d-flex">
22                         <Button onClickAction={() => {onButtonClickAction(card)}} buttonText={!(inCart) ? "Buy Now" : "In Cart: Add +1"}/>
23                         <span className="icon">
24                         <FontAwesomeIcon icon={(inWishlist || isHovered) ? fasHeart : farHeart}  onClick={() => {onIconClickAction(card)}}  onMouseOver={()
25                         </span>
26                     </div>
27                 </div>
28             }
29         </div>
30     )
31 }
32 export default Card
```

This component has a prop called extended. And if that prop's value is true it shows additional information about the card that is being displayed, as well as an interactable button and wishlist icon (functions also need to be passed down as a prop).

When used to display the featured cards in the homepage, extended prop is passed false and we just see the {card.image} with it's design as given in card.css

Here is how it looks on /cards page.
Again onButtonClickAction and onIconClickAction needs to be passed from App.js downwards to make them interactable.

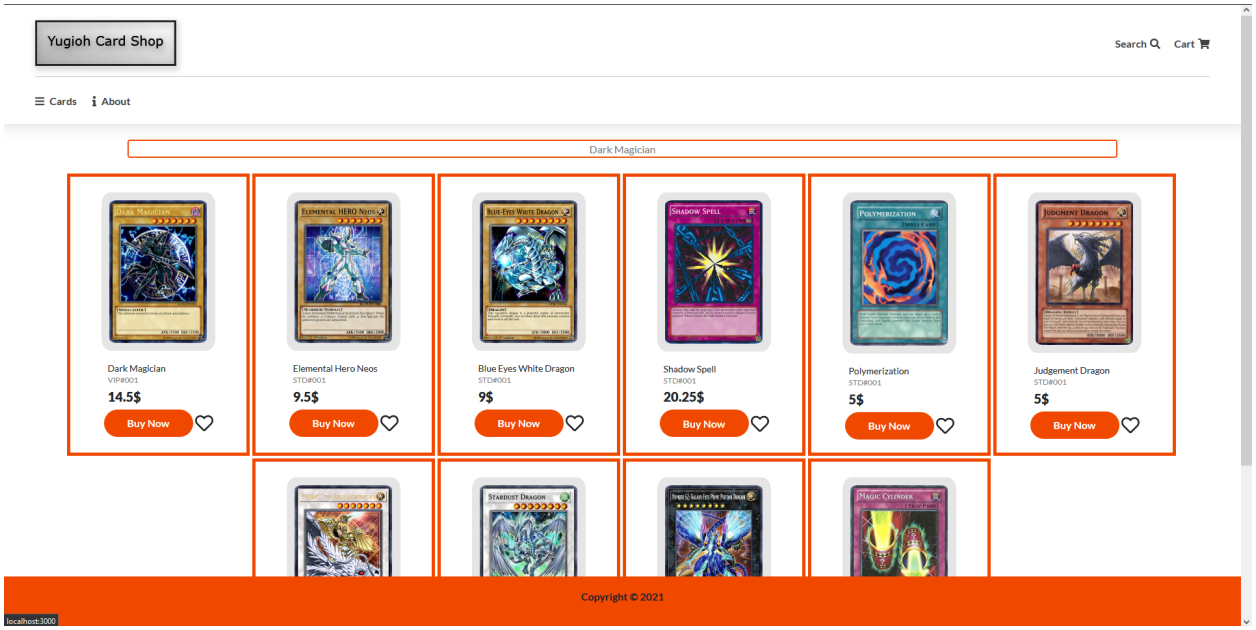
I also passed the "inList" function as a prop which helps us to know if a card is in a wishlist or in a cart. If it is already in the cart, Button's text changes accordingly.

States of the wishlist Icon

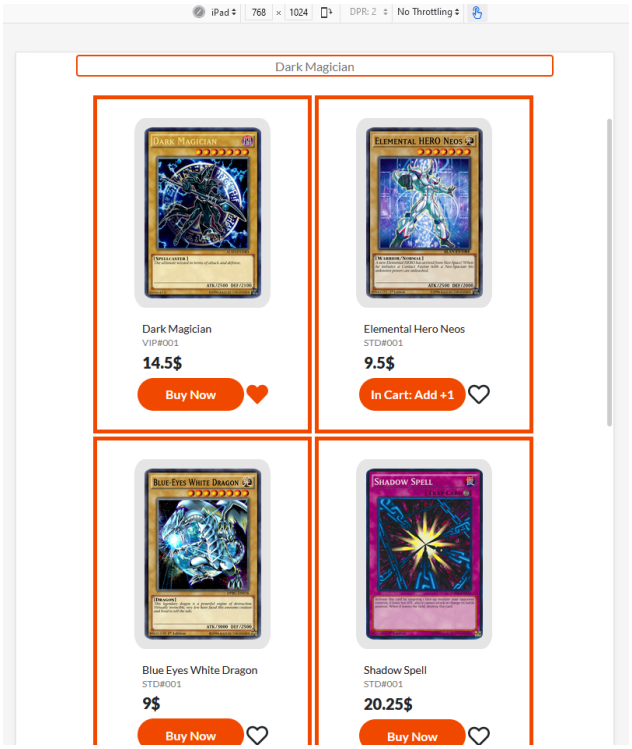
: Default state.

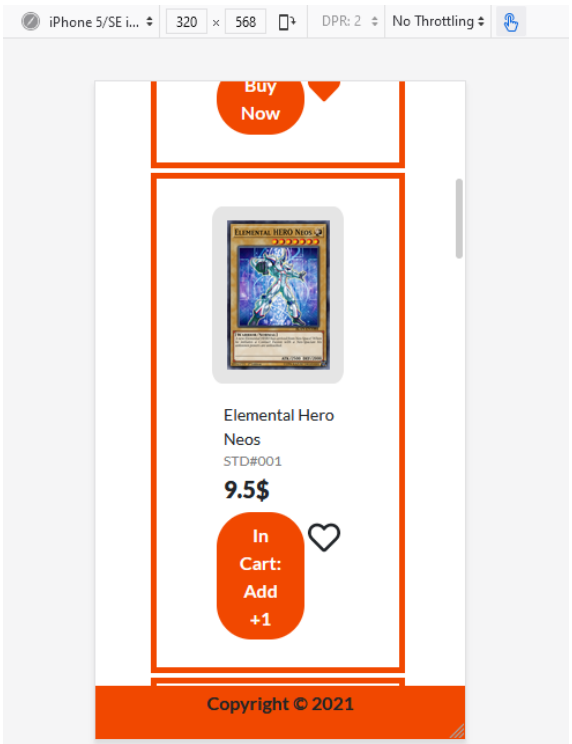
: Hovered state.

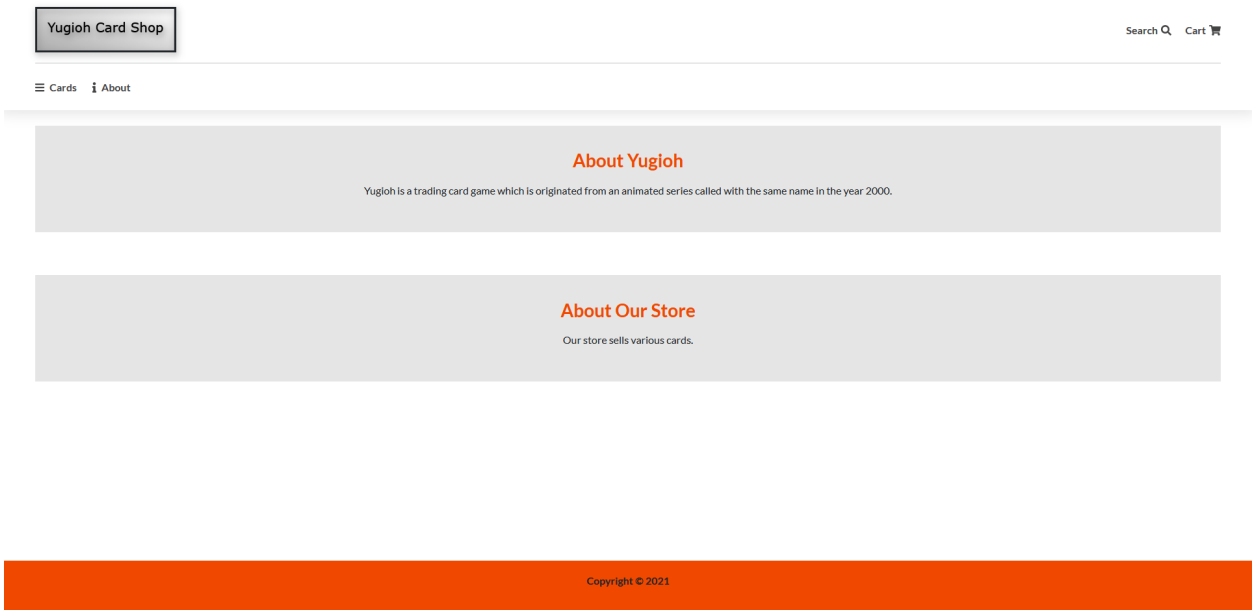
: Wishlisted state.



Above is the /cards page. Here card components are wrapped with “product-wrapper” which gives them borders, also each one of the products are flex items. There is a small search bar above the items. Sadly I could only get it to work on exact searches, but case insensitive nonetheless.







About page is very empty sadly. I didn't know what to do with this page.

I want to talk about the functions and variables I wrote before I move on with the PageShoppingCart, the last page on the site.

```
const cards = [ {
  id: 0,
  name: "Dark Magician",
  edition: "VIP#001",
  cardImage: "https://i.pinimg.com/originals/c9/43/93/c94393b38644cf7b9becb800cd335115.jpg",
  price: 14.50,
},
{
  id: 1,
  name: "Elemental Hero Neos",
  edition: "STD#001",
  cardImage: "https://images-wixmp-ed30a86b8c4ca887773594c2.wixmp.com/f/627fe721-846f-4f75
  price: 9.50,
},
{
  id: 2,
  name: "Blue Eyes White Dragon",
  edition: "STD#001",
  cardImage: "https://i.pinimg.com/originals/2a/31/e4/2a31e48bafe34f7692cfd5f63fd1310a.png",
  price: 9.00,
},
```

This is a constant list. This is not the list that is being iterated when displaying the cards. Since I couldn't implement the database into my work I made a new variable that doesn't change whatever happens and used it as my refresher whenever the user tried to refresh the card search result in /cards page.

```
const [displayCards, setDisplayCards] = useState([...cards])
```

This is the list that is being iterated for /books page.

```
const handleSearch = (event) => {
    if(event.key === 'Enter'){
      setDisplayCards([...cards])
      if(event.target.value != ""){
        setDisplayCards(displayCards.filter((card) => card.name.toLowerCase() === event.target.value.toLowerCase()))
        event.target.value = ""
      }
    }
}
```

This function is passed to PageCards and used in onKeyPress inside the search-box item in the page. If the pressed key is ‘Enter’ it refreshes the displayed cards then filters out anything besides the entered card, modifying the displayedCards.

```
const [cartItems, setCartItems] = useState([])
const [wishlistedItems, setWishlistedItems] = useState([])
```

First variable holds the items in the cart and the second holds the wishlisted items.

```
const inList = (card, fromCart) => {
  if (fromCart){
    var copy = [...cartItems];
  }
  else{
    var copy = [...wishlistedItems];
  }
  copy = copy.filter((item) => item.id === card.id);
  if(copy.length == 0)
    return false;
  return true;
}
```

This function takes two parameters. First parameter is the card object and the second is boolean that determines whether we are searching in wishlistedItems or cartItems. If fromCart is true then we are making our search in inCartItems and so on.

If the copy variable has length of zero then that means the card is not contained in the list we are searching and it returns false. Else it returns true meaning it exists in our searched list.

```
const addToWishlist  = (card) => {
  if(!inList(card, false))
    setWishlistedItems([...wishlistedItems, card])
  else{
    setWishlistedItems(wishlistedItems.filter((item) => item.id !== card.id))
  }
}
```

This function adds a card to a wishlist or removes it from the list if it is already in the wishlist.

```
const addToCart = (card) => {
  if(!inList(card, true))
    setCartItems([...cartItems, {...card, amount:1}]);
  else{
    setCartItems(cartItems.map((item) => (item.id === card.id ? {...item, amount: item.amount+1} : item)  ))
  }
}
```

This function adds a card to the cartItems. If the card already exists in the cartItems, it increases the "amount" attribute by one else it creates a new object {...card, amount:1} and appends it to the existing cartItems.

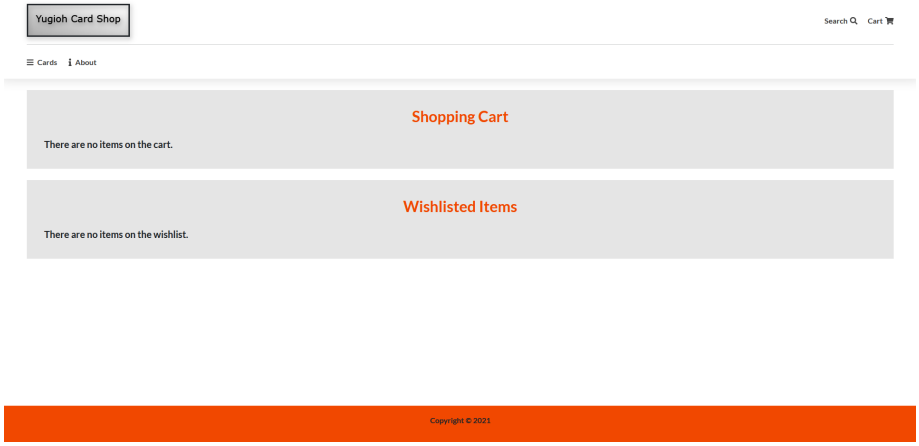
```
const removeFromCart = (card) => {
  if(card.amount == 1)
    setCartItems(cartItems.filter((item) => item.id !== card.id));
  else{
    setCartItems(cartItems.map((item) => (item.id === card.id ? {...item, amount: item.amount-1} : item)  ))
  }
}
```

This function is to remove an item from the cart. If the amount is 1, it removes the card object from the list. If it's > 1 it just decreases the amount by one. To be able to call this function an item already must be inside the cartItems so we don't need to use the inList function here.

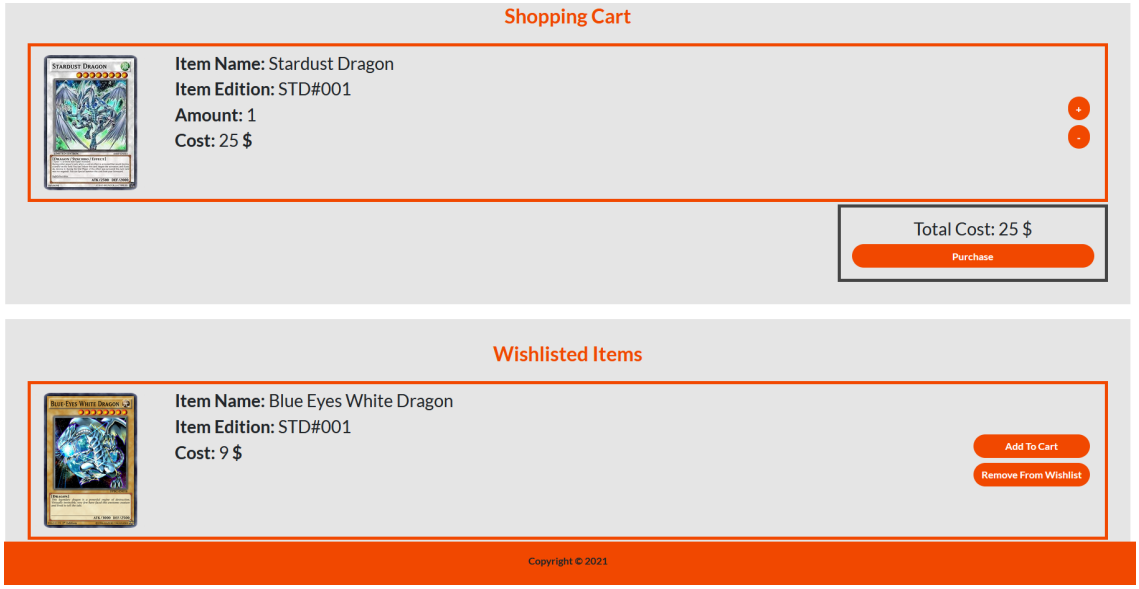
```
const completePurchase = () => {
  alert("Purchase has been made.")
  /* Empty cart items after the purchase has been made. */
  setCartItems([])
}
```

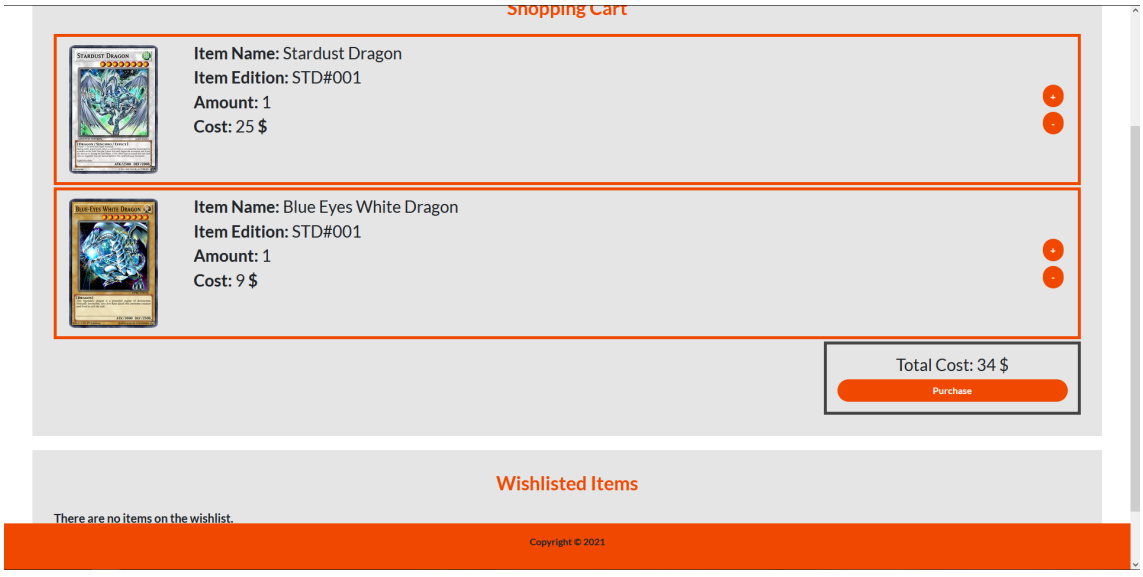
Sadly this is all I could do for the purchase function, I couldn't implement backend and database to my project. This function just pops up an alert and clears the cartItems.

PageShoppingCart -- With no items:



PageShoppingCart -- With items:





I will show local functions in the PageShoppingCart:

```
const costs = cartItems.map((item) => {
    return item.amount * item.price
})

function add(accumulator, a) {
  return accumulator + a;
}
```

These two are used together later to get the total costs of the items in the cartItems. First function returns a list of floats, then second is used to sum those float values.

```
18    const displayCart = cartItems.map((item) => {
19      return (
20        <div key={item.id} className="product-wrapper d-flex px-2 mb-1">
21          <div className="image-section">
22            <Card card={item} extended={false} />
23          </div>
24          <div className="item-info d-flex flex-column flex-grow-1 py-2 px-5">
25            <h3><b>Item Name:</b> {item.name} </h3>
26            <h3><b>Item Edition:</b> {item.edition}</h3>
27            <h3><b>Amount:</b> {item.amount}</h3>
28            <h3><b>Cost:</b> {item.amount * item.price} <b>$</b></h3>
29          </div>
30          <div className="amount-modifier d-flex flex-column justify-content-center px-3">
31          <Button buttonText="+" onClickAction={() => {addToCart(item)}}/>
32          <div className="mb-2"></div>
33          <Button buttonText="-" onClickAction={() => {removeFromCart(item) }}/>
34          </div>
35
36        </div>
37      );
38    });
```

This function is run for each item inside the cartItems. I wrapped it in a product-wrapper the same one I used in card.css, I made it a d-flex that consisted of 3 columns. I gave the product-wrapper div padding with “px-2” and gave a bottom margin 1 using “mb-1”. So each product is separated by a margin.

1. section is the image section, it is the image of the card that is being purchased, so I called Card component with extended prop = false.

2. section is a flex container in of itself. It is a flex-column that consists of 4 rows. I made it so that it fills the remaining area using “flex-grow-1”. I displayed information about the item here.
3. Section is again a flex container that displays buttons to modify the amount. I centered the buttons using “justify-content-center” and placed buttons with corresponding functionalities inside. One increases the amount of the item by one and the other decreases the item amount by one. If the new amount is 0 item is removed from the cart.

```
const displayWishlisted = wishlistedItems.map((item) => {
 return (
  <div key={item.id} className="product-wrapper d-flex px-2 mb-1">
    <div className="image-section">
     <Card card={item} extended={false} />
    </div>
    <div className="item-info d-flex flex-column flex-grow-1 py-2 px-5">
     <h3><b>Item Name:</b> {item.name} </h3>
     <h3><b>Item Edition:</b> {item.edition}</h3>
     <h3><b>Cost:</b> {item.price} <b>$</b></h3>
    </div>
    <div className="amount-modifier d-flex flex-column justify-content-center px-3">
    <Button buttonText="Add To Cart" onClickAction={() => {addToCart(item); addToWishlist(item)}}/>
    <div className="mb-2"></div>
    <Button buttonText="Remove From Wishlist" onClickAction={() => {addToWishlist(item)}}/>
    </div>

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

This is a function with the same concept, but to display wishlisted items.

```
62    return (
63      <MainLayout>
64      <div className="main py-4 px-5">
65
66        <div className="shopping-cart d-flex flex-column mb-4">
67          <h2 className="head-section">Shopping Cart</h2>
68          {cartItems.length>0 ? displayCart : <p className="text">There are no items on the cart.</p>}
69          {cartItems.length>0 &&
70          <div className=".clickable-main d-flex justify-content-end">
71            <div className="clickable-section d-flex flex-column justify-content-end">
72            <h3>Total Cost: {costs.reduce(add,0)} $</h3>
73            <Button buttonText="Purchase" onClickAction={purchaseAction}/>
74            </div>
75          </div>
76          }
77        </div>
78
79        <div className="wishlisted-items mb-4">
80          <h2 className="head-section">Wishlisted Items</h2>
81          {wishlistedItems.length>0 ? displayWishlisted : <p className="text">There are no items on the wishlist.</p>}
82        </div>
83
84      </div>
85      </MainLayout>
86    );
87  }
88
89  export default PageShoppingCart;
```

This is the return value of cart.js (PageShoppingCart) I wrapped everything inside a class called main and moved its position, matching it to the main layout of the site. Here I separated the remaining part to two divs each having margin bottom - 4. First is for cartItems. “head-section” prints the title of the box. Centers the text, making them bold and giving them color. Then if there exists items inside “cartItems” it calls displayCart else it just shows a “text” “saying there are no items on the cart”
Then if there exists items in the cart, it prints out a “clickable-main” that is justified to the end of the div. That has width of 25%
This is also a flex item consisting of two rows. One is showing total cost using the functions I showed earlier, then a “Purchase” button.

“wishlisted-items” section is similar if items exist, it displays, but with different buttons allowing the user to add items to the cart or removing the items from the wishlist.

Project Showcase

https://drive.google.com/file/d/1AwsTCl5M1R3BWZac7jb0_qZZ3Lif4RxD/view?usp=sharing