Question: Share your logic behind adding the "CartItem" React element in cart/Cart.jsx for the Burger Shop's Cart page. How did you structure the attributes like title, image (Img), and implement the increment and decrement counters for the new item "Cheese Burger with French Fries"?

Solution:

The logic behind adding the "CartItem" React element in Cart.jsx for the Burger Shop's Cart page is about creating a reusable component that represents an item in the shopping cart. This component is responsible for displaying the item's title, image, and quantity, as well as providing buttons to increment and decrement the quantity.

```
const CartItem = ({ value, title, img, increment, decrement }) => (
  <div className="cartItem">
    <div>
      <h4>{title}</h4>
      <img src={img} alt="Item" />
    </div>

    <div>
      <button onClick={decrement}>-</button>
      <input type="number" readOnly value={value} />
      <button onClick={increment}>+</button>
    </div>
  </div>
);
```

CartItem is a functional component that takes several props: value, title, img, increment, and decrement.

The component renders the item's title and image in one div, and another div contains buttons to increment and decrement the item quantity.

The increment and decrement functions are passed as props and are used to update the quantity in the parent Cart component.

```
    <CartItem
      title={"Cheese Burger with French Fries"}
      img={burger3}
      value={cartItems[3]}
      increment={() => increment(3)}
      decrement={() => decrement(3)}
    />
```

The Cart component renders multiple instances of the CartItem component for different items in the cart.

For the "Cheese Burger with French Fries" item, the title, img, value, increment, and decrement props are passed with specific values. I have it in the code above

The increment and decrement functions are arrow functions that call the respective functions in the Cart component with the item ID (in this case, 3).

```
const increment = (itemId) => {
  setCartItems((prevItems) => ({
    ...prevItems,
    [itemId]: prevItems[itemId] + 1,
  }));
};

const decrement = (itemId) => {
  if (cartItems[itemId] > 0) {
    setCartItems((prevItems) => ({
      ...prevItems,
      [itemId]: prevItems[itemId] - 1,
    }));
  }
};
```

A. The increment function updates the quantity of the specified item by incrementing it by 1.
B. The decrement function checks if the quantity is greater than 0 before decrementing it by 1 to avoid negative quantities.

```
const taxRate = 0.1; // 10%

const shippingCharges = 0;//I will set this to zero for now, change it to 50 later, delete this later

const subtotal = Object.keys(cartItems).reduce(
  (acc, itemId) => acc + cartItems[itemId] * itemPrices[itemId],
  0
);

const tax = subtotal * taxRate;
const total = subtotal + tax + shippingCharges;
```

Good to mention that the tax, subtotal, shippingcharges and total will be updated accordingly.

Question: Share your coding logic employed in cart/Shipping.jsx to generate a dropdown menu containing country names using the getAllCountries method and map function. How did you set the value attribute and display country names in the dropdown menu?

Solution:

In the Shipping.jsx file, I've implemented the logic to create a dropdown menu containing country names using the getAllCountries method from the country-state-city library.

```
{/* COUNTRY DROPDOWN */}
<label>Country</label>
<select onChange={handleCountryChange} value={selectedCountry}>
  <option value="">Country</option>
  {Country.getAllCountries().map((country) => (
    <option value={country.isoCode} key={country.isoCode}>
      {country.name}
    </option>
  ))}
</select>
```

Explanation:

1. State Initialization:

```
const [selectedCountry, setSelectedCountry] = useState("");
```

The selectedCountry state is used to keep track of the currently selected country in the dropdown.

2. Event Handler for Country Change:

```
const handleCountryChange = (e) => {
  const countryIsoCode = e.target.value;
  setSelectedCountry(countryIsoCode);
  setSelectedState(""); // Reset selected state when country changes
};
```

The handleCountryChange function is triggered when the user selects a country from the dropdown. It updates the selectedCountry state and resets the selectedState state.

3. Country Dropdown Markup:

```
{/* COUNTRY DROPDOWN */}
<label>Country</label>
```

```
<select onChange={handleCountryChange} value={selectedCountry}>
  <option value="">Country</option>
  {Country.getAllCountries().map((country) => (
    <option value={country.isoCode} key={country.isoCode}>
      {country.name}
    </option>
  ))}
</select>
```

- The <select> element represents the dropdown menu for selecting the country.
- The onChange is set to the handleCountryChange function, making sure that the state is updated when the user selects a country.
- The value attribute is set to selectedCountry, which it is responsive to the currently selected country in the dropdown.

4. Mapping Through Countries:

```
  {Country.getAllCountries().map((country) => (
    <option value={country.isoCode} key={country.isoCode}>
      {country.name}
    </option>
  ))}
```

- The Country.getAllCountries().map method is used to iterate through the list of countries.
- For each country, an <option> element is created in the dropdown with the value attribute set to
  the country's ISO code and the country name as the displayed content.
- The key attribute is set to the country's ISO code to ensure uniqueness.
- I got help from StackOverflow source https://stackoverflow.com/questions/53850252/reactjs-country-data-map

Question: Illustrate your logic behind implementing a <Popup> element to generate a "Order Placed" message on clicking the "Confirm Order" button. How did you manage the CSS elements for styling the Popup?

Solution:

1. State Initialization:

```
 const [orderPlaced, setOrderPlaced] = useState(false);
```

The orderPlaced state is used to control whether the Popup should be displayed or not.

2. Popup Component:

```
        <Popup open={orderPlaced} position="right center">
          <div style={{ color: "red", position: "absolute", top: "50%",
right: "100%", transform: "translateY(-50%)", backgroundColor: "#fff",
padding: "10px", borderRadius: "5px", boxShadow: "0 0 10px rgba(0, 0, 0, 0.2)"
}}>
            Order Placed successfully!
          </div>
        </Popup>
```

The <Popup> component is rendered conditionally based on the orderPlaced state.

The position prop is set to "right center" to position the Popup on the right side of the screen at the center vertically.

The inner <div> contains the message "Order Placed successfully!" and is styled using inline CSS.

3. Popup Visibility Control:

```
<Popup open={orderPlaced} position="right center">
      <code-here>
</Popup>
```

The open prop of the <Popup> component is set to the orderPlaced state, making sure that the Popup is displayed when orderPlaced is true and hidden when orderPlaced is false.

I have used the reactjs-popup library to create a Popup that appears on successful order placement. The CSS styling makes sure that the Popup is located and styled in a visually good way.