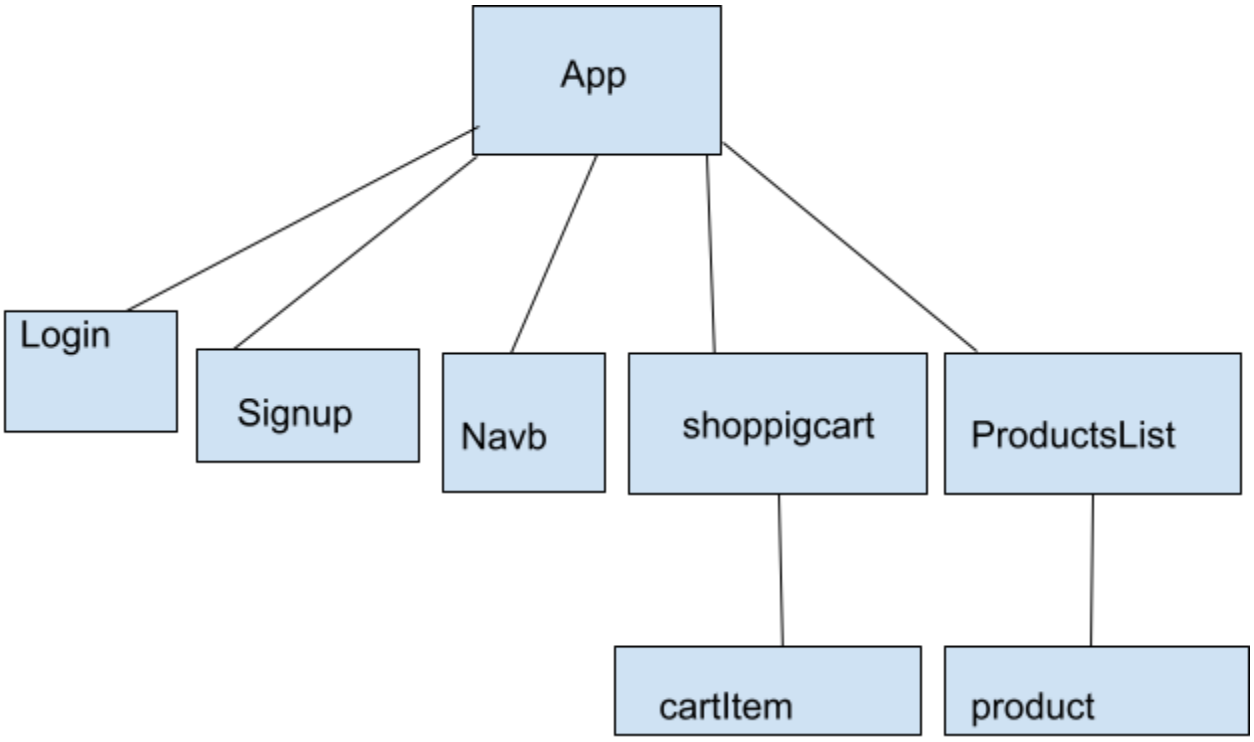
Shopping cart react project

Our application has the following structure:



Login: login component has a form with username and password field and submit button. On clicking submit button we call login function which calls loginapp function from App(it sends request to django using product-services.js) if server approves the request then it saves user and token in state and local storage:

```
setToken(response.data.access);
setUser(user.username);
localStorage.setItem('token', response.data.access);
localStorage.setItem('user', user.username);
After that we go to / which shows list of products
```

Signup: signup component has a form with several fields and submit button. On clicking submit button we call signup function which calls singuppapp function from App(it sends request to django using product-services.js) if server approves the request then it creates new user and we save user and token in state and local storage:

```
setToken(response.data.access);
setUser(user.username);
localStorage.setItem('token', response.data.access);
```

```
localStorage.setItem('user', user.username);
After that we go to / which shows list of products
```

Navb: Navb component has a navbar section with following links /, /login, /signup. It also has a product search field and search button on clicking the button list of products are filtered based on search term in a search field. It also has an icon/button to open the shopping cart.

ShoppingCart: for shopping cart we use `Offcanvas` from react-bootstrap. It gets following properties from App :`{isOpen, closeCart, cartItems, setCartItems, token}`
It shows total price , pay button and list of cart items using cartItem component. By clicking on pay button it sends request to django to updated is_paid property to true using `async function payCart() {`

```
   await ProductDataService.payCart({"is_paid": true}, token)
}
```

We can also remove cartItem from the shopping cart using the button. And the following function is called

```
async function removeFromCart(id) {
   const existingProductIndex = cartItems.findIndex(item => item.product.id ===
id)
   await ProductDataService.deleteCartItem(cartItems[existingProductIndex].id,
token)
   setCartItems(prevCartItems => prevCartItems.filter(item => item.product.id
!== id))
}
```

CartItem: CartItem component uses `horizontal` Stack to show details of cart item. It includes image or product, name, quantity,price,total price

ProductsList: ProductsList component uses `Row` of bootstrap to show list of products in rows: it receives following properties from app:`{token, cartItems, setCartItems, products, cartId}`

```
<Row md={2} xs={1} lg={3} className='g-3'>

{products.map((product) => {
   return (<Product key={product.id} product={product} cartItems={cartItems}
                    setCartItems={setCartItems} token={token} cartId={cartId}/>
   )
}
   )}
   </Row>)}
```

It goes over each product and created Product component and puts in a Row

Product: product component shows details of a product. It gets following properties from productsList `{product, cartItems, setCartItems, token, cartId}`
It shows the image and name of the product and quantity if the user is logged in and the product is in cart. For logged in users it shows an add to cart button to add the item to cart if the product is not inside the cart. If the product is already inside the cart it shows the quantity of the product in the cart and +, - and remove buttons, they increase, decrease and delete product from the cart respectively. Respective functions update state and send requests to django using ProductsService