Redux Toolkit Panier

Référence : https://react-redux.js.org/

Dépendances *:*

```
npm install axios

npm install react-router-dom

npm install react-redux @reduxjs/toolkit

npm i bootstrap react-bootstrap

npm i react-toastify

npm install @mui/material @emotion/react @emotion/styled
```

Remarque : Ajouter l'option **--force** en cas ou l'installation échoue

Ajouter l'appel à bootstrap dans le fichier index.js

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Fichier src/Axios/Api.js

```
import axios from 'axios'

export default axios.create({
baseURL:"https://ecommercebackend2023j.vercel.app/api"
})
```

Création de services Frontend

Fichier src/services/ArticleService.js

```
import Api from "../Axios/Api";
```

```
const ARTICLE_API="/articles"
export const fetchArticles=async()=> {
return await Api.get(ARTICLE_API);
}
export const fetchArticleById=async(articleId)=> {
return await Api.get(ARTICLE_API + '/' + articleId);
}
export const deleteArticle=async(articleId) =>{
return await Api.delete(ARTICLE_API + '/' + articleId);
}
export const addArticle=async(article)=> {
return await Api.post(ARTICLE_API, article);
}
export const editArticle=(article) =>{
return Api.put(ARTICLE_API + '/' + article._id, article);
}
```

CategorieService.js

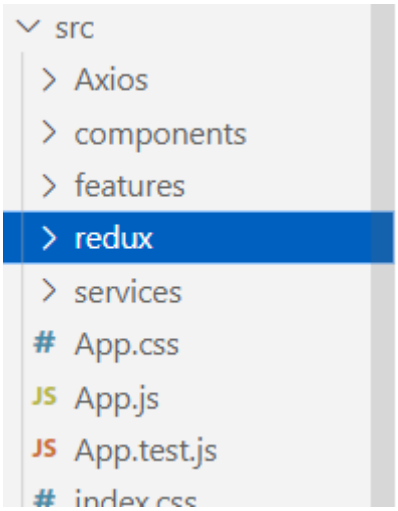
```
import Api from "../Axios/Api";
const CATEGORIE_API="/categories"
export const fetchCategories=async()=> {
return await Api.get(CATEGORIE_API);
}
export const fetchCategorieById=async(categorieId)=> {
return await Api.get(CATEGORIE_API + '/' + categorieId);
}
export const deleteCategorie=async(categorieId) =>{
return await Api.delete(CATEGORIE_API + '/' + categorieId);
}
export const addCategorie=async(categorie)=> {
return await Api.post(CATEGORIE_API,categorie);
}
export const editCategorie=(categorie) =>{
return Api.put(CATEGORIE_API + '/' + categorie._id, categorie);
}
```

ScategorieService.js

```
import Api from "../Axios/Api";
const SCATEGORIE_API="/scategories"
export const fetchSCategories=async()=> {
return await Api.get(SCATEGORIE_API);
}
export const fetchSCategorieById=async(scategorieId)=> {
return await Api.get(SCATEGORIE_API + '/' + scategorieId);
}
```

```
const fetchSCategorieByCat=(idcat) =>{return Api.get(SCATEGORIE_API + '/cat/' +
idcat);
}
export const deleteSCategorie=async(scategorieId) =>{
return await Api.delete(SCATEGORIE_API + '/' + scategorieId);
}
export const addSCategorie=async(scategorie)=> {
return await Api.post(SCATEGORIE_API,scategorie);
}
export const editSCategorie=(scategorie) =>{
return Api.put(SCATEGORIE_API + '/' + scategorie._id, scategorie);
}
```

Ajouter les dossiers features et redux comme indiqué ci-dessous



Créer les slices (Redux toolkit)

Pour chaque classe on crée les slices des Cruds

Fichier src/features/articleSlice.js

```
import { createSlice,createAsyncThunk } from '@reduxjs/toolkit'
import {fetchArticles,addArticle,deleteArticle,editArticle,fetchArticleById} from
"../services/ArticleService"

export const getArticles = createAsyncThunk(
    "article/getArticles",
    async (_, thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
      try {
      const res = await fetchArticles();
```

```
    return res.data;
    }
    catch (error) {
      return rejectWithValue(error.message);
    }
  }
);

export const createArticle = createAsyncThunk(
  "article/createArticle",
  async (article, thunkAPI) => {
    const { rejectWithValue } = thunkAPI;
    try{
    const res= await addArticle(article);
    return res.data
  }
  catch (error) {
    return rejectWithValue(error.message);
  }
  }
);

export const delArticle = createAsyncThunk(
  "article/delArticle",
  async (id,thunkAPI) => {
    const { rejectWithValue } = thunkAPI;
    try{
    await deleteArticle(id);
    return  id ;
  }
  catch (error) {
    return rejectWithValue(error.message);
  }
  });

export const updateArticle = createAsyncThunk(
    "article/updateArticle",
    async (article, thunkAPI) => {
    const { rejectWithValue } = thunkAPI;
      try{
     const res= await editArticle(article);
      return res.data

    }
    catch (error) {
      return rejectWithValue(error.message);
    }

    }
```

```
    );

  export const findArticleByID = createAsyncThunk(
    "article/findArticleByID",
    async (id,thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
      try{
      const res = await fetchArticleById(id);
      return res.data;
    }
    catch (error) {
      return rejectWithValue(error.message);
    }
    });
export const articleSlice = createSlice({
  name: 'article',
  initialState:{
    articles:[],
    article:{},
    isLoading: false,
    success:null,
    error:null,
  },


  extraReducers: (builder) => {
    //get articles
    builder
    .addCase(getArticles.pending, (state, action) => {

      state.isLoading=true;
      state.error=null;
    })
    .addCase(getArticles.fulfilled, (state, action) => {
      state.isLoading=false;
      state.error = null;
      state.articles=action.payload;
    })
    .addCase(getArticles.rejected, (state, action) => {
      state.isLoading=false;
      state.error=action.payload;
      console.log("impossible de se connecter au serveur")
    })

    //insertion article
    .addCase(createArticle.pending, (state, action) => {
      state.isLoading=true;
      state.error=null;
      state.success=null;
```

```
  })
  .addCase(createArticle.fulfilled, (state, action) => {

    state.articles.push(action.payload);
    state.isLoading=false;
    state.error=null;
    state.success=action.payload;
  })
  .addCase(createArticle.rejected, (state, action) => {
    state.isLoading=false;
    state.error=action.payload;
    state.success=null;
  })
  //Modification article
  .addCase(updateArticle.pending, (state, action) => {
    state.isLoading=true;
    state.error=null;
    state.success=null;
  })
  .addCase(updateArticle.fulfilled, (state, action) => {
    state.articles = state.articles.map((item) =>
    item._id === action.payload._id ? action.payload : item
  );
state.isLoading=false;
state.error=null;
state.success=action.payload;
   })
  //Delete article
  .addCase(delArticle.pending, (state, action) => {
    state.isLoading=true;
    state.error=null;
  })
  .addCase(delArticle.fulfilled, (state, action) => {
    state.isLoading=false;
    state.error=null;
    state.articles=state.articles.filter((item)=> item._id!==action.payload)

  })
  .addCase(delArticle.rejected, (state, action) => {
    state.isLoading=false;
    state.error=action.payload;
  })
//Fectch article
  .addCase(findArticleByID.pending, (state, action) => {
    state.isLoading = true
    state.error=null;

    })
  .addCase(
```

```
    findArticleByID.fulfilled,(state, action) => {
      state.isLoading = false
      state.error = null
      state.article=action.payload;
   })

  }

  }

)

export default articleSlice.reducer;
```

Fichier src/features/scategorieSlice.js

```
import { createSlice,createAsyncThunk } from '@reduxjs/toolkit'
import
{fetchSCategories,addSCategorie,deleteSCategorie,editSCategorie,fetchSCategorieById
} from "../services/ScategorieService"

export const getScategories = createAsyncThunk(
    "scategorie/getScategories",
    async (_, thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
      try {
      const res = await fetchSCategories();
      return res.data;
      }
      catch (error) {
        return rejectWithValue(error.message);
      }
    }
  );

  export const createScategorie = createAsyncThunk(
    "scategorie/createScategorie",
    async (scategorie, thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
      try{
      const res= await addSCategorie(scategorie);
      return res.data
    }
    catch (error) {
      return rejectWithValue(error.message);
```

```
    }
    }
  );

  export const deleteScategorie = createAsyncThunk(
    "scategorie/deleteScategorie",
    async (id,thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
      try{
      await deleteSCategorie(id);
      return  id ;
    }
    catch (error) {
      return rejectWithValue(error.message);
    }
    });

  export const updateScategorie = createAsyncThunk(
      "scategorie/updateScategorie",
      async (scategorie, thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
        try{
       const res= await editSCategorie(scategorie);
        return res.data

      }
      catch (error) {
        return rejectWithValue(error.message);
      }

      }
    );

  export const findScategorieByID = createAsyncThunk(
    "scategorie/findScategorieByID",
    async (id,thunkAPI) => {
      const { rejectWithValue } = thunkAPI;
      try{
      const res = await fetchSCategorieById(id);
      return res.data;
    }
    catch (error) {
      return rejectWithValue(error.message);
    }
    });
export const scategorieSlice = createSlice({
  name: 'scategorie',
  initialState:{
    scategories:[],
```

```
  scategorie:{},
  isLoading: false,
  success:null,
  error:null,
},


extraReducers: (builder) => {
  //get scategories
  builder
  .addCase(getScategories.pending, (state, action) => {

    state.isLoading=true;
    state.error=null;
  })
  .addCase(getScategories.fulfilled, (state, action) => {
    state.isLoading=false;
    state.error = null;
    state.scategories=action.payload;
  })
  .addCase(getScategories.rejected, (state, action) => {
    state.isLoading=false;
    state.error=action.payload;
    console.log("impossible de se connecter au serveur")
  })

  //insertion scategorie
  .addCase(createScategorie.pending, (state, action) => {
    state.isLoading=true;
    state.error=null;
    state.success=null;
  })
  .addCase(createScategorie.fulfilled, (state, action) => {

    state.scategories.push(action.payload);
    state.isLoading=false;
    state.error=null;
    state.success=action.payload;
  })
  .addCase(createScategorie.rejected, (state, action) => {
    state.isLoading=false;
    state.error=action.payload;
    state.success=null;
  })
  //Modification scategorie
  .addCase(updateScategorie.pending, (state, action) => {
    state.isLoading=true;
    state.error=null;
    state.success=null;
```

```
    })
    .addCase(updateScategorie.fulfilled, (state, action) => {
      state.scategories = state.scategories.map((item) =>
      item._id === action.payload._id ? action.payload : item
    );
  state.isLoading=false;
  state.error=null;
  state.success=action.payload;
     })
    //Delete scategorie
    .addCase(deleteScategorie.pending, (state, action) => {
      state.isLoading=true;
      state.error=null;
    })
    .addCase(deleteScategorie.fulfilled, (state, action) => {
      state.isLoading=false;
      state.error=null;
      state.scategories=state.scategories.filter((item)=>
item._id!==action.payload)

    })
    .addCase(deleteScategorie.rejected, (state, action) => {
      state.isLoading=false;
      state.error=action.payload;
    })
  //Fectch scategorie
    .addCase(findScategorieByID.pending, (state, action) => {
      state.isLoading = true
      state.error=null;

      })
    .addCase(
     findScategorieByID.fulfilled,(state, action) => {
      state.isLoading = false
      state.error = null
      state.scategorie=action.payload;
   })

  }

  }

)


export default scategorieSlice.reducer;
```

Créer le dossier redux sous src

Créer le Fichier src/redux/store.js


```
import { configureStore } from '@reduxjs/toolkit'
import articlesReducer from "../features/articleSlice"
import scategoriesReducer from "../features/scategorieSlice"

const store = configureStore({
  reducer: {
    storearticles:articlesReducer,
    storescategories: scategoriesReducer,
  }
})
export default store
```

Importer le store dans le fichier index.js

Fichier src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import 'bootstrap/dist/css/bootstrap.min.css';
import reportWebVitals from './reportWebVitals';
import store from './redux/store';
import { Provider } from 'react-redux' ;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store={store}>

    <App />

    </Provider>
  </React.StrictMode>
);
```

```
// 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();
```

Gestion des composants Articles

Fichier src/components/articlesRedux/Listarticles.js

```
import React, { useEffect,useCallback } from "react";
import { useDispatch } from "react-redux";
import {getArticles} from "../../features/articleSlice";
import AfficheArticles from "./AfficheArticles";

const Listarticles = () => {


    const dispatch = useDispatch();

    const initFetch = useCallback(() => {
        dispatch(getArticles());
      }, [dispatch])

      useEffect(() => {
        initFetch()
      }, [initFetch])


  return (
    <div>
              <AfficheArticles/>
    </div>
  )
}

export default Listarticles
```

Installer react-loading

Site : https://www.npmjs.com/package/react-loading

```
npm i react-loading --force
```

Fichier src/components/articlesRedux/AfficheArticles.js

```
import React from 'react'
import Card from '@mui/material/Card';
import CardActions from '@mui/material/CardActions';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
import ReactLoading from 'react-loading';
import {useSelector} from "react-redux"


const AfficheArticles = () => {

    const {articles,isLoading,error} = useSelector((state)=>state.storearticles);


const renderArticles = () => {
    if (isLoading) return  <center><ReactLoading type='spokes' color="red"
height={'8%'} width={'8%'}  /></center>
    if (error) return <p>Impossible d'afficher la liste des articles...</p>

    return <React.Fragment>

    {articles  &&
          <div
style={{"display":"flex","flexWrap":"wrap","justifyContent":"left"}}>
          {articles.map((art,ind)=>{
              return  <Card sx={{ maxWidth: 'auto',margin: 1 }} key={ind}>
              <CardMedia
                component="img"
                alt="image"
                height="160"
                image={art.imageart}
              />
              <CardContent>
                <Typography gutterBottom variant="h6" component="div">
                {art.reference}
                </Typography>
                <Typography variant="body2" color="text.secondary">
                Prix : {art.prix} DT
                </Typography>
              </CardContent>
              <CardActions>
              <Button disabled={art.qtestock<=1}
                  variant="contained" color="secondary" size="large"
```

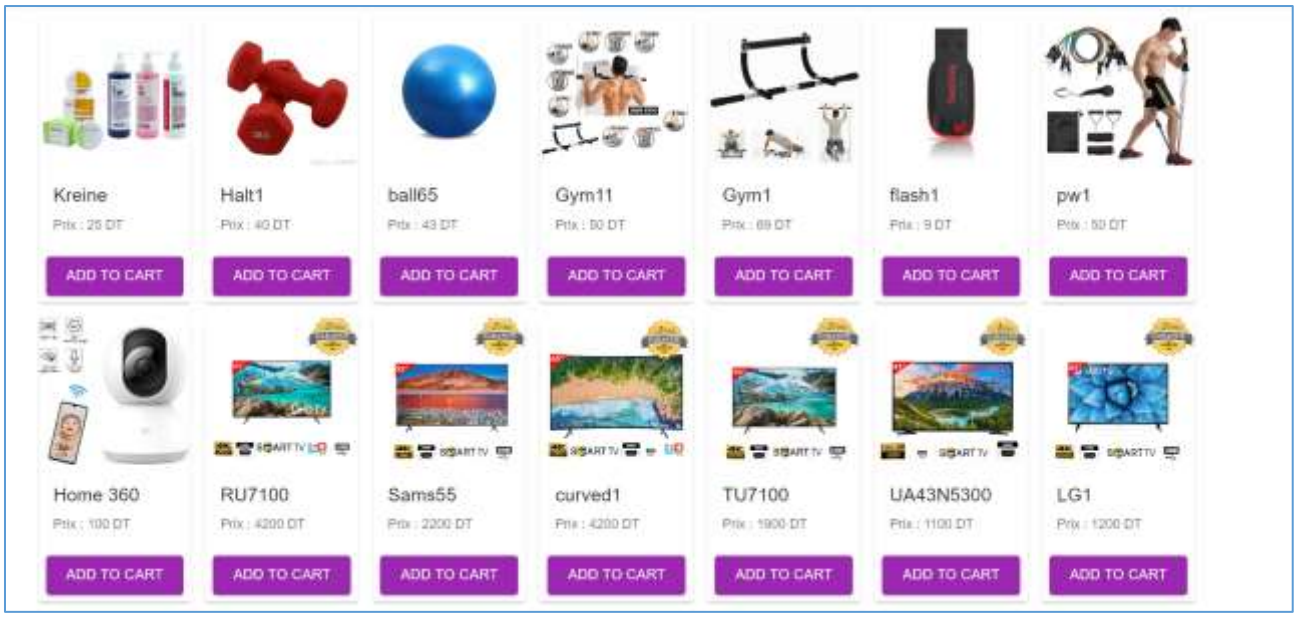
```
                >
                 {art.qtestock<=1? "OUT OF SOLD": "Add to cart"}
          </Button>

              </CardActions>
            </Card>

        })}</div>
      }
  </React.Fragment>
  }


  return (
    <div className="container">
        {renderArticles()}
   </div>

  )
}

export default AfficheArticles
```

Fichier src/App.js

```
import Listarticles from './components/articlesRedux/Listarticles';
```

…

```
<Route path='/' element={<Listarticles/>}/>
```



Gestion des paniers

Fichier App.css

```
.bag-quantity {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 25px;
  width: 25px;
  border-radius: 50%;
  background: yellow;
  font-size: 14px;
  font-weight: 700;
  color: black;
  margin-left: 5px;
}

/* Home */

.home-container {
  padding: 2rem 4rem;
}
.home-container h2 {
  font-size: 40px;
  font-weight: 400;
  text-align: center;
}
```

```css
.products {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  margin-top: 2rem;
}
.product {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  margin: 1rem auto;
  padding: 1rem;
  border-radius: 15px;
  width: 250px;
  max-width: 100%;
  height: 400px;
  box-shadow: -5px -5px 10px rgba(255, 255, 255, 0.5),
    2px 2px 5px rgba(94, 104, 121, 0.3);
}
.product h3 {
  font-size: 25px;
  font-weight: 400;
}
.product img {
  width: 80%;
  margin-top: 1rem;
  margin-left: auto;
  margin-right: auto;
}
.product .details {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
.product .details .price {
  font-size: 20px;
  font-weight: 700;
}
.product button {
  width: 100%;
  height: 40px;
  border-radius: 5px;
  margin-top: 2rem;
  font-weight: 400;
  letter-spacing: 1.15px;
  background-color: #4b70e2;
  color: #f9f9f9;
  border: none;
  outline: none;
```

```
  cursor: pointer;
}

/* Cart */
.cart-container {
  padding: 2rem 4rem;
}
.cart-container h2 {
  font-weight: 400;
  font-size: 30px;
  text-align: center;
}
.cart-container .titles {
  margin: 2rem 0 1rem 0;
}
.cart-container .titles h3 {
  font-size: 14px;
  font-weight: 400;
  text-transform: uppercase;
}
.cart-item,
.cart-container .titles {
  display: grid;
  align-items: center;
  grid-template-columns: 3fr 1fr 1fr 1fr;
  column-gap: 0.5rem;
}
.cart-item {
  border-top: 1px solid rgb(187, 187, 187);
  padding: 1rem 0;
}
.cart-container .titles .product-title {
  padding-left: 0.5rem;
}
.cart-container .titles .total {
  padding-right: 0.5rem;
  justify-self: right;
}
.cart-item .cart-product {
  display: flex;
}
.cart-item .cart-product img {
  width: 100px;
  max-width: 100%;
  margin-right: 1rem;
}
.cart-item .cart-product h3 {
  font-weight: 400;
}
```

```css
.cart-item .cart-product button {
  border: none;
  outline: none;
  margin-top: 0.7rem;
  cursor: pointer;
  background: none;
  color: gray;
}
.cart-item .cart-product button:hover {
  color: black;
}

.cart-item .cart-product-quantity {
  display: flex;
  align-items: flex-start;
  justify-content: center;
  width: 130px;
  max-width: 100%;
  border: 0.5px solid rgb(177, 177, 177);
  border-radius: 5px;
}
.cart-item .cart-product-quantity button {
  border: none;
  outline: none;
  background: none;
  padding: 0.7rem 1.5rem;
  cursor: pointer;
}
.cart-item .cart-product-quantity .count {
  padding: 0.7rem 0;
}
.cart-item .cart-product-total-price {
  padding-right: 0.5rem;
  justify-self: right;
  font-weight: 700;
}

/* cart summary */
.cart-summary {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  border-top: 1px solid rgb(187, 187, 187);
  padding-top: 2rem;
}
.cart-summary .clear-btn {
  width: 130px;
  height: 40px;
  border-radius: 5px;
```

```
  font-weight: 400;
  letter-spacing: 1.15px;
  border: 0.5px solid rgb(177, 177, 177);
  color: gray;
  background: none;
  outline: none;
  cursor: pointer;
}
.cart-checkout {
  width: 270px;
  max-width: 100%;
}
.cart-checkout .subtotal {
  display: flex;
  justify-content: space-between;
  font-size: 20px;
}
.cart-checkout .amount {
  font-weight: 700;
}
.cart-checkout p {
  font-size: 14px;
  font-weight: 200;
  margin: 0.5rem 0;
}
.cart-checkout button {
  width: 100%;
  height: 40px;
  border-radius: 5px;
  font-weight: 400;
  letter-spacing: 1.15px;
  background-color: #4b70e2;
  color: #f9f9f9;
  border: none;
  outline: none;
  cursor: pointer;
}
.continue-shopping,
.start-shopping {
  margin-top: 1rem;
}
.continue-shopping a,
.start-shopping a {
  color: gray;
  text-decoration: none;
  display: flex;
  align-items: center;
}
.continue-shopping a span,
```

```
.start-shopping a span {
  margin-left: 0.5rem;
}
.cart-empty {
  font-size: 20px;
  margin-top: 2rem;
  color: rgb(84, 84, 84);
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

Remarque : Faire appel à ce fichier dans App.js

```
import './App.css';
```

Fichier src/features/cartSlice.js

```
import { createSlice } from "@reduxjs/toolkit";
import { toast } from "react-toastify";

const initialState = {
  cartItems: localStorage.getItem("cartItems")
    ? JSON.parse(localStorage.getItem("cartItems"))
    : [],
  cartTotalQuantity: 0,
  cartTotalAmount: 0,
};

const cartSlice = createSlice({
  name: "cart",
  initialState,
  reducers: {
    addToCart(state, action) {
      const existingIndex = state.cartItems.findIndex(
        (item) => item._id === action.payload._id
      );

      if (existingIndex >= 0) {
        state.cartItems[existingIndex] = {
          ...state.cartItems[existingIndex],
          cartQuantity: state.cartItems[existingIndex].cartQuantity + 1,
        };
        toast.info("Increased product quantity", {
          position: "bottom-left",
        });
      } else {
```

```
      let tempProductItem = { ...action.payload, cartQuantity: 1 };
      state.cartItems.push(tempProductItem);
      toast.success("Product added to cart", {
        position: "bottom-left",
      });

    }
    localStorage.setItem("cartItems", JSON.stringify(state.cartItems));
  },
  decreaseCart(state, action) {
    const itemIndex = state.cartItems.findIndex(
      (item) => item._id === action.payload._id
    );

    if (state.cartItems[itemIndex].cartQuantity > 1) {
      state.cartItems[itemIndex].cartQuantity -= 1;

      toast.info("Decreased product quantity", {
        position: "bottom-left",
      });
    } else if (state.cartItems[itemIndex].cartQuantity === 1) {
      const nextCartItems = state.cartItems.filter(
        (item) => item._id !== action.payload._id
      );

      state.cartItems = nextCartItems;

      toast.error("Product removed from cart", {
        position: "bottom-left",
      });
    }

    localStorage.setItem("cartItems", JSON.stringify(state.cartItems));
  },
  removeFromCart(state, action) {
    state.cartItems.map((cartItem) => {
      if (cartItem._id === action.payload._id) {
        const nextCartItems = state.cartItems.filter(
          (item) => item._id !== cartItem._id
        );

        state.cartItems = nextCartItems;

        toast.error("Product removed from cart", {
          position: "bottom-left",
        });
      }
      localStorage.setItem("cartItems", JSON.stringify(state.cartItems));
      return state;
```

```
      });
    },
    getTotals(state, action) {
      let { total, quantity } = state.cartItems.reduce(
        (cartTotal, cartItem) => {
          const { prix, cartQuantity } = cartItem;
          const itemTotal = prix * cartQuantity;

          cartTotal.total += itemTotal;
          cartTotal.quantity += cartQuantity;

          return cartTotal;
        },
        {
          total: 0,
          quantity: 0,
        }
      );
      total = parseFloat(total.toFixed(2));
      state.cartTotalQuantity = quantity;
      state.cartTotalAmount = total;
    },
    clearCart(state, action) {
      state.cartItems = [];
      localStorage.setItem("cartItems", JSON.stringify(state.cartItems));
      toast.error("Cart cleared", { position: "bottom-left" });
    },
  },
});

export const { addToCart, decreaseCart, removeFromCart, getTotals, clearCart } =
  cartSlice.actions;

export default cartSlice.reducer;
```

Fichier src/redux/store.js

```
import { configureStore } from '@reduxjs/toolkit'
import articlesReducer from "../features/articleSlice"
import scategoriesReducer from "../features/scategorieSlice"
import cartSliceReducer from "../features/cartSlice"

const store = configureStore({
  reducer: {
    storearticles:articlesReducer,
    storescategories: scategoriesReducer,
```

```
    storecart:cartSliceReducer,
  }
})
export default store
```

Fichier src/components/articlesRedux/Cart.js

La function .toFixed(3) pour fixer 3 chiffres après la virgule pour les totaux

```
import React,{useEffect,useCallback} from 'react'
import { useDispatch, useSelector } from "react-redux";
import {
  addToCart,
  clearCart,
  decreaseCart,
  getTotals,
  removeFromCart,
} from "../../features/cartSlice";

import { Link } from "react-router-dom";

const Cart = () => {

    const cart = useSelector((state) => state.storecart);
    const dispatch = useDispatch();

    const computeTotal = useCallback(() => {
      dispatch(getTotals());
    }, [cart, dispatch])

    useEffect(() => {
      computeTotal()
    }, [computeTotal])


    const handleAddToCart  = useCallback((product) => {
      dispatch(addToCart(product));
    }, [dispatch])

    const handleDecreaseCart = useCallback((product) => {
      dispatch(decreaseCart(product));
    }, [dispatch])

    const handleRemoveFromCart = useCallback((product) => {
```

```
    dispatch(removeFromCart(product));
  }, [dispatch])

  const handleClearCart = useCallback(() => {
    dispatch(clearCart());
  }, [dispatch])

return (
  <div className="cart-container">
    <h2>Shopping Cart</h2>
    {cart.cartItems.length === 0 ? (
      <div className="cart-empty">
        <p>Panier Vide</p>
        <div className="start-shopping">
          <Link to="/">

            <span>Start Shopping</span>
          </Link>
        </div>
      </div>
    ) : (
      <div>
        <div className="titles">
          <h3 className="product-title">Product</h3>
          <h3 className="price">Price</h3>
          <h3 className="quantity">Quantity</h3>
          <h3 className="total">Total</h3>
        </div>
        <div className="cart-items">
          {cart.cartItems &&
            cart.cartItems.map((cartItem) => (
              <div className="cart-item" key={cartItem._id}>
                <div className="cart-product">
                 <img src={`${cartItem.imageart}`} alt={cartItem.designation}/>
                  <div>
                    <h3>{cartItem.designation}</h3>
                    <p>{cartItem.reference}</p>
                    <button onClick={() => handleRemoveFromCart(cartItem)}>
                      Remove
                    </button>
                  </div>
                </div>
                <div className="cart-product-price"> {cartItem.prix} TND</div>
                <div className="cart-product-quantity">
                  <button onClick={() => handleDecreaseCart(cartItem)}>
                    -
                  </button>
                  <div className="count">{cartItem.cartQuantity}</div>
                  <button onClick={() => handleAddToCart(cartItem)}>+</button>
```

```
                  </div>
                  <div className="cart-product-total-price">
                    {(cartItem.prix * cartItem.cartQuantity).toFixed(3)} TND
                  </div>
                </div>
              ))}
          </div>
          <div className="cart-summary">
            <button className="clear-btn" onClick={() => handleClearCart()}>
              Clear Cart
            </button>
            <div className="cart-checkout">
              <div className="subtotal">
                <span>Subtotal</span>
                <span className="amount">{cart.cartTotalAmount.toFixed(3)}
TND</span>
              </div>
              <p>Taxes and shipping calculated at checkout</p>
              <button>Check out</button>
              <div className="continue-shopping">
                <Link to="/">

                  <span>Continue Shopping</span>
                </Link>
              </div>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

export default Cart;
```

Fichier src/components/articlesRedux/AfficheArticles.js

```
import React from 'react'

import Card from '@mui/material/Card';
import CardActions from '@mui/material/CardActions';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
```

```
import ReactLoading from 'react-loading';

import { useDispatch, useSelector } from "react-redux";
import { addToCart } from "../../features/cartSlice";

import { useNavigate } from "react-router-dom";

const AfficheArticles = () => {

    const {articles,isLoading,error} = useSelector((state)=>state.storearticles);

    const dispatch = useDispatch();
    let navigate=useNavigate();


const handleAddToCart = (art) => {
      dispatch(addToCart(art));
      navigate("/cart");
    };


const renderArticles = () => {
    if (isLoading) return  <center><ReactLoading type='spokes' color="red"
height={'8%'} width={'8%'}  /></center>
    if (error) return <p>Impossible d'afficher la liste des articles...</p>

    return <React.Fragment>

    {articles  &&
          <div
style={{"display":"flex","flexWrap":"wrap","justifyContent":"left"}}>
          {articles.map((art,ind)=>{
              return  <Card sx={{ maxWidth: 'auto',margin: 1 }} key={ind}>
              <CardMedia
                component="img"
                alt="image"
                height="160"
                image={art.imageart}
              />
              <CardContent>s
                <Typography gutterBottom variant="h6" component="div">
                {art.reference}
                </Typography>
                <Typography variant="body2" color="text.secondary">
                Prix : {art.prix} DT
                </Typography>
              </CardContent>
              <CardActions>
```

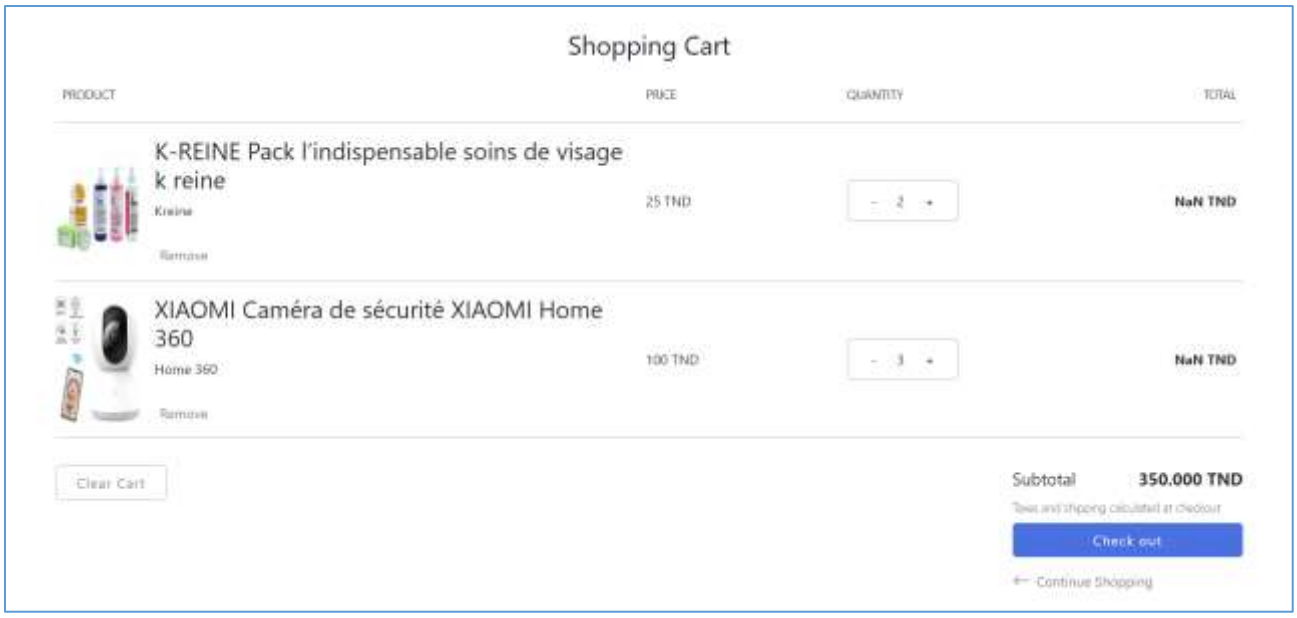
```
              <Button disabled={art.qtestock<=1} onClick={() =>
handleAddToCart(art)}
                  variant="contained" color="secondary" size="large"

                >
                 {art.qtestock<=1? "OUT OF SOLD": "Add to cart"}
          </Button>

              </CardActions>
            </Card>

        })}</div>
      }
  </React.Fragment>
  }


  return (
    <div className="container">
        {renderArticles()}
   </div>

  )
}

export default AfficheArticles
```

Fichier src/App.js

```
import Cart from './components/articlesRedux/Cart';
```

…

```
<Route path='/cart' element={<Cart/>}/>
```



Fichier src/components/NavScrolls.jsx

Ajouter le panier et afficher les quantités dans le navbar

```
import React from 'react'
import Button from 'react-bootstrap/Button';
import Container from 'react-bootstrap/Container';
import Form from 'react-bootstrap/Form';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';
import {Link } from "react-router-dom"

import Badge from '@mui/material/Badge';
import IconButton from '@mui/material/IconButton';
import ShoppingCartIcon from '@mui/icons-material/ShoppingCart';
import { useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";

const NavScroll = () => {

  const navigate = useNavigate();
  const {cartTotalQuantity} = useSelector((state) => state.storecart);

  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container fluid>
```

```
          <IconButton size="large"
             edge="end"
             aria-label="account of current user"

             aria-haspopup="true"

             color="error"
             onClick={()=>{navigate("/cart")}}
             >

             <ShoppingCartIcon sx={{ fontSize: 40 }}/>
             <Badge badgeContent={cartTotalQuantity>0?cartTotalQuantity:0}
color="success">
               </Badge>
             </IconButton>
```

```
        <Navbar.Brand href="#"> </Navbar.Brand>
        <Navbar.Toggle aria-controls="navbarScroll" />
        <Navbar.Collapse id="navbarScroll">
          <Nav
            className="me-auto my-2 my-lg-0"
            style={{ maxHeight: '100px' }}
            navbarScroll
          >
            <Nav.Link  as={Link} to="/articles">Articles</Nav.Link>
            <Nav.Link as={Link} to="/categories">Catégories</Nav.Link>
            <Nav.Link as={Link} to="/scategories">Sous Catégories</Nav.Link>
            <NavDropdown title="Link" id="navbarScrollingDropdown">
              <NavDropdown.Item href="#action3">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action4">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action5">
                Something else here
              </NavDropdown.Item>
            </NavDropdown>
            <Nav.Link href="#" disabled>
              Link
            </Nav.Link>
          </Nav>
          <Form className="d-flex">
            <Form.Control
              type="search"
              placeholder="Search"
              className="me-2"
              aria-label="Search"
            />
            <Button variant="outline-success">Search</Button>
```

```
          </Form>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  )
}

export default NavScroll
```