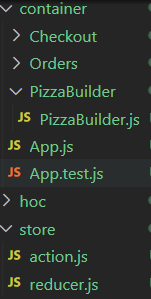
# INSTALLATION OF REDUX AND REDUX-REACT

npm install –save redux react- redux

Create store folder:



Actions.js

export const ADD\_INGREDIENTS = 'ADD\_INGREDIENTS';

export const REMOVE\_INGREDIENTS = 'REMOVE\_INGREDIENTS';

reducer.js

import \* as actionTypes from './action';

const initialState= {

    ingredients: null,

    totalPrice: 4,

}

const Reducer = (state=initialState,actions) => {

    // switch(){

    //     case(actionTypes.ADD\_INGREDIENTS):

    // }

    return state;

}

export default Reducer;

InitialState is taken from PizzaBuilder.js.

const PizzaBuilder = (props) => {

    const [state, setstate] = useState({

        ingredients: null,

        totalPrice: 4,

    });

    const [purchases, setpurchases] = useState({

        purchasable: false,

        purchasing: false

    })

# BASIC REDUX SETUP

Index.js

const app = (

  <Provider>

    <BrowserRouter>

      <App/>

    </BrowserRouter>

  </Provider>

);

Finally after creating store to import reducer

import App from './container/App';

import reportWebVitals from './reportWebVitals';

import {BrowserRouter} from 'react-router-dom'

import { Provider } from 'react-redux';

import {createStore} from 'redux';

import Reducer from './store/reducer';

const store = createStore(Reducer);

const app = (

  <Provider store={store}>

    <BrowserRouter>

      <App />

    </BrowserRouter>

  </Provider>

);

# FINISIHING THE REDUCER FOR INGREDIENTS

NOTE:

…state = copy old state

We are overriding new ingredients to given ingredients. //we can do it by ES6 special syntax to dynamically override property in a given JS object

[action.ingredientName] : state.ingredients[action.ingredientName] + 1

ingredientName = property name(as you desire) expected to get this on action

state.ingredients[action.ingredientName] = new value

import \* as actionTypes from './action';

const initialState= {

    ingredients: {

        salad: 0,

        bacon: 0,

        cheese: 0,

        meat: 0

    },

    totalPrice: 4,

}

const Reducer = (state=initialState,action) => {

    switch(action.type){

        case actionTypes.ADD\_INGREDIENTS:

            return{

                //copying old state

                ...state,

                ingredients: {

                    ...state.ingredients,

                    //override with new ingredients

                    //we can do it by ES6 special syntax to dynamically override property in a given JS object

                    [action.ingredientName] :

state.ingredients[action.ingredientName] + 1

                }

            };

        case actionTypes.REMOVE\_INGREDIENTS:

            return{

                //copying old state

                ...state,

                ingredients: {

                    ...state.ingredients,

                    [action.ingredientName] : s

tate.ingredients[action.ingredientName] - 1

                }

            };

        default:

            return state;

    }

}

export default Reducer;

# CONNECTING THE PIZZA BUILDER CONTAINER TO OUR STORE:

PizzaBuilder.js

import  { connect } from 'react-redux'

import \* as actionTypes from '../../store/action';

import React,{useState} from 'react';

import Aux from '../../hoc/Auxilliary/Auxilliary';

import Pizza from '../../components/Pizza/Pizza';

import BuildControls from '../../components/Pizza/BuildControls/BuildControls';

import Modal from '../../components/UI/Modal/Modal';

import OrderSummary from '../../components/Pizza/OrderSummary/OrderSummary';;

const INGREDIENTS\_PRICES = {

    salad:0.5,

    cheese: 0.4,

    meat: 1,

    bacon: 2,

}

const PizzaBuilder = (props) => {

    const [state, setstate] = useState({

        totalPrice: 4,

    });

    const [purchases, setpurchases] = useState({

        purchasable: false,

        purchasing: false

    })

    function updatePurchaseState(ingredients) {

        const sum = Object.keys( ingredients )

        .map( igKey => {

            return ingredients[igKey];

        } )

        .reduce( ( sum, el ) => {

            return sum + el;

        }, 0 );

        setpurchases( { purchasable: sum > 0 } );

    }

    const addIngredientsHandler = (type) => {

        const oldCount = state.ingredients[type];

        const updatedCount = oldCount +1;

        const updatedIngredients ={

            ...state.ingredients

        };

        updatedIngredients[type] =updatedCount;

        const priceAddition = INGREDIENTS\_PRICES[type];

        const oldPrice= state.totalPrice;

        const newPrice = oldPrice + priceAddition;

        setstate({ingredients: updatedIngredients, totalPrice: newPrice});

        updatePurchaseState(updatedIngredients);

    }

    const removeIngredientsHandler = (type) => {

        const oldCount = state.ingredients[type];

        if( oldCount <= 0) {

            return;

        }

        const updatedCount = oldCount -1;

        const updatedIngredients ={

            ...state.ingredients

        };

        updatedIngredients[type] =updatedCount;

        const priceDeduction = INGREDIENTS\_PRICES[type];

        const oldPrice= state.totalPrice;

        const newPrice = oldPrice - priceDeduction;

        setstate({ingredients: updatedIngredients, totalPrice: newPrice});

        // PurchasesUpdate();

    }

    const purchaseHandler = () => {

        setpurchases({purchasing: true});

    }

    const purchasedCancelHandler = () => {

        setpurchases({purchasing:false})

    }

    const purcharseContinueHandler = () => {

        // console.log(props);

        const queryParams= []; //empty array

        for( let i in state.ingredients){

            queryParams.push(encodeURIComponent(i) + '=' + encodeURIComponent(state.ingredients[i]));

        }

        const queryString = queryParams.join('&');

        props.history.push({

            pathname: '/checkout',

            search: '?'+queryString,

        });

    }

    const disableInfo = {

        ...props.ings

    }

    for( let key in disableInfo){

        disableInfo[key] = disableInfo[key] <=0

    }//{salad: true, meat:false, ...}

    return(

        <Aux>

            <Modal show={purchases.purchasing}

modalClosed={purchasedCancelHandler}>

                <OrderSummary

                price={state.totalPrice}

                ingredients={props.ings}

                purchaseCancel={purchasedCancelHandler}

                purchaseContinue ={purcharseContinueHandler}

                />

            </Modal>

             <Pizza ingredients={props.ings}/>

             <BuildControls

             removeIngredient= {props.onIngredientRemoved}

             moreIngredient={props.onIngredientAdded}

             disabled={disableInfo}

             ordered = {purchaseHandler}

             purchasabled = {purchases.purchasable}

             price={state.totalPrice}

             />

        </Aux>

    );

}

const mapStateToProps = state =>{

    return{

        ings: state.ingredients

    };

}

const mapDispatchToProps = dispatch => {

    return{

        onIngredientAdded: (ingName) => dispatch({

            type: actionTypes.ADD\_INGREDIENTS,

            ingredientName: ingName

        }),

        onIngredientRemoved: (ingName) => dispatch({

            type: actionTypes.REMOVE\_INGREDIENTS,

//[action.ingredientName] : state.ingredients[action.ingredientName] + 1

            ingredientName: ingName

        })

    }

}

export default connect(mapStateToProps,mapDispatchToProps)(PizzaBuilder);

NOTE:

props.onIngredientRemoved : yesma argument kina nahaleko vanda chai, BuildControls ma already ctrl.type vanera pass garisakeko cha

import React from 'react';

import './BuildControls.css';

import BuildControl from './BuildControl/BuildControl';

const controls = [

    {label: 'Salad', type: 'salad'},

    {label: 'Bacon', type: 'bacon'},

    {label: 'Cheese', type: 'cheese'},

    {label: 'Meat', type: 'meat'},

];

const BuildControls = (props) => (

    <div className='BuildControls'>

    <p>Current Price: <strong>{props.price.toFixed(2)}</strong></p>

         { controls.map(ctrl =>(

             <BuildControl

             key={ctrl.label}

             label={ctrl.label}

            //  type={ctrl.type}

            removed={() => props.removeIngredient(ctrl.type)}

            //kun type chai add garne vanera type ctrl props bata pathako

             added={() => props.moreIngredient(ctrl.type)}

             disabled ={props.disabled[ctrl.type]}

             />

         ))}

         <button

         disabled={!props.purchasabled}

         onClick={props.ordered}

         >ORDER NOW</button>

    </div>

);

export default BuildControls;

# WORKING ON THE TOTAL PRICE CALCULATION

Reducer.js

import \* as actionTypes from './action';

const initialState= {

    ingredients: {

        salad: 0,

        bacon: 0,

        cheese: 0,

        meat: 0

    },

    totalPrice: 4,

}

const INGREDIENTS\_PRICES = {

    salad:0.5,

    cheese: 0.4,

    meat: 1,

    bacon: 2,

}

const Reducer = (state=initialState,action) => {

    switch(action.type){

        case actionTypes.ADD\_INGREDIENTS:

            return{

                ...state,

                ingredients: {

                    ...state.ingredients,

                    [action.ingredientName] : state.ingredients[action.ingredientName] + 1

                },

                totalPrice: state.totalPrice + INGREDIENTS\_PRICES[action.ingredientName]

            };

        case actionTypes.REMOVE\_INGREDIENTS:

            return{

                ...state,

                ingredients: {

                    ...state.ingredients,

                    [action.ingredientName] : state.ingredients[action.ingredientName] - 1

                },

                totalPrice: state.totalPrice - INGREDIENTS\_PRICES[action.ingredientName]

            };

        default:

            return state;

    }

}

export default Reducer;

PizzaBuilder.js

import  { connect } from 'react-redux'

import \* as actionTypes from '../../store/action';

import React,{useState} from 'react';

import Aux from '../../hoc/Auxilliary/Auxilliary';

import Pizza from '../../components/Pizza/Pizza';

import BuildControls from '../../components/Pizza/BuildControls/BuildControls';

import Modal from '../../components/UI/Modal/Modal';

import OrderSummary from '../../components/Pizza/OrderSummary/OrderSummary';

const PizzaBuilder = (props) => {

    const [purchases, setpurchases] = useState({

        purchasable: false,

        purchasing: false

    })

    function updatePurchaseState(ingredients) {

        const sum = Object.keys( ingredients )

        .map( igKey => {

            return ingredients[igKey];

        } )

        .reduce( ( sum, el ) => {

            return sum + el;

        }, 0 );

        setpurchases( { purchasable: sum > 0 } );

    }

    const purchaseHandler = () => {

        setpurchases({purchasing: true});

    }

    const purchasedCancelHandler = () => {

        setpurchases({purchasing:false})

    }

    const purcharseContinueHandler = () => {

        // console.log(props);

        const queryParams= []; //empty array

        for( let i in props.ings){

            queryParams.push(encodeURIComponent(i) + '=' + encodeURIComponent(props.ings[i]));

        }

        const queryString = queryParams.join('&');

        props.history.push({

            pathname: '/checkout',

            search: '?'+queryString,

        });

    }

    const disableInfo = {

        ...props.ings

    }

    for( let key in disableInfo){

        disableInfo[key] = disableInfo[key] <=0

    }//{salad: true, meat:false, ...}

    return(

        <Aux>

            <Modal show={purchases.purchasing} modalClosed={purchasedCancelHandler}>

                <OrderSummary

                price={props.price}

                ingredients={props.ings}

                purchaseCancel={purchasedCancelHandler}

                purchaseContinue ={purcharseContinueHandler}

                />

            </Modal>

             <Pizza ingredients={props.ings}/>

             <BuildControls

             removeIngredient= {props.onIngredientRemoved}

             moreIngredient={props.onIngredientAdded}

             disabled={disableInfo}

             ordered = {purchaseHandler}

             purchasabled = {purchases.purchasable}

             price={props.price}

             />

        </Aux>

    );

}

const mapStateToProps = state =>{

    return{

        ings: state.ingredients,

        price: state.totalPrice

    };

}

const mapDispatchToProps = dispatch => {

    return{

        onIngredientAdded: (ingName) => dispatch({

            type: actionTypes.ADD\_INGREDIENTS,

            ingredientName: ingName

        }),

        onIngredientRemoved: (ingName) => dispatch({

            type: actionTypes.REMOVE\_INGREDIENTS,

            ingredientName: ingName

        })

    }

}

export default connect(mapStateToProps,mapDispatchToProps)(PizzaBuilder);

# REDUX UI AND STATE [order button disable problem]:

ONE WAY (LOCAL UI STATE):

PizzaBuilder.js

const PizzaBuilder = (props) => {

    const [purchases, setpurchases] = useState({

        purchasing: false

    })

    function updatePurchaseState(ingredients) {

        const sum = Object.keys( ingredients )

        .map( igKey => {

            return ingredients[igKey];

        } )

        .reduce( ( sum, el ) => {

            return sum + el;

        }, 0 );

        return sum > 0 ; //returns true or false after comparing ingredients

    }

    const purchaseHandler = () => {

        setpurchases({purchasing: true});

    }

    const purchasedCancelHandler = () => {

        setpurchases({purchasing:false})

    }

    const purcharseContinueHandler = () => {

        // console.log(props);

        const queryParams= []; //empty array

        for( let i in props.ings){

            queryParams.push(encodeURIComponent(i) + '=' + encodeURIComponent(props.ings[i]));

        }

        const queryString = queryParams.join('&');

        props.history.push({

            pathname: '/checkout',

            search: '?'+queryString,

        });

    }

    const disableInfo = {

        ...props.ings

    }

    for( let key in disableInfo){

        disableInfo[key] = disableInfo[key] <=0

    }//{salad: true, meat:false, ...}

    return(

        <Aux>

            <Modal show={purchases.purchasing} modalClosed={purchasedCancelHandler}>

                <OrderSummary

                price={props.price}

                ingredients={props.ings}

                purchaseCancel={purchasedCancelHandler}

                purchaseContinue ={purcharseContinueHandler}

                />

            </Modal>

             <Pizza ingredients={props.ings}/>

             <BuildControls

             removeIngredient= {props.onIngredientRemoved}

             moreIngredient={props.onIngredientAdded}

             disabled={disableInfo}

             ordered = {purchaseHandler}

//removing state and adding updatePurchaseState function with argument ingredients which is passed through reducer

             purchasabled = {updatePurchaseState(props.ings)}

             price={props.price}

             />

        </Aux>

    );

}

ANOTHER WAY (REDUX STATE)

# AJUSTING CHECKOUT AND CONTACT DATA