Contents

[UNDERSTANDING AUTHENTICATION IN SINGLE PAGE APPLICATIONS 2](#_Toc78124532)

[REQUIRED ADJUSTMENTS 2](#_Toc78124533)

[ADDING AN AUTH FORM 3](#_Toc78124534)

[ADDING ACTIONS [submit button action] 6](#_Toc78124535)

[GETTING A TOKEN FROM THE BACKEND 11](#_Toc78124536)

[ADDING SIGN-IN 15](#_Toc78124537)

[STORING THE TOKEN 21](#_Toc78124538)

[Logging users out 27](#_Toc78124539)

[ACCESSING PROTECTED RESOURCES [access to data based on authentication] 31](#_Toc78124540)

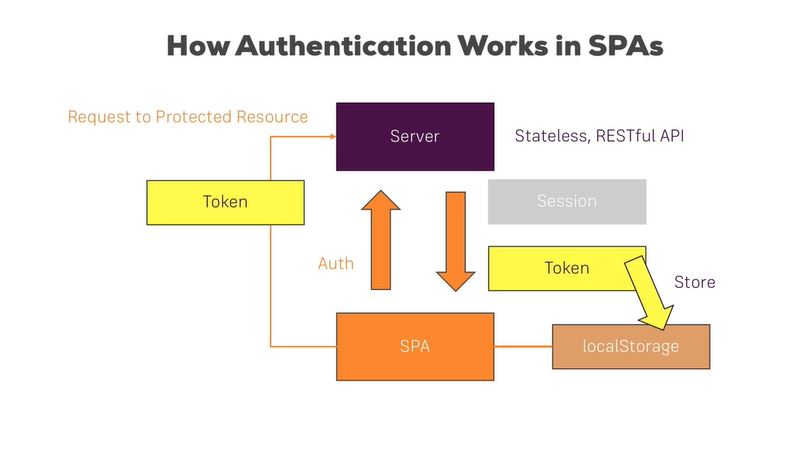
[UPDATING THE UI DEPENDING ON UI STATE [showing logout in navigation]: 35](#_Toc78124541)

[ADDING LOGOUT LINK [logout and redirect to home page] 41](#_Toc78124542)

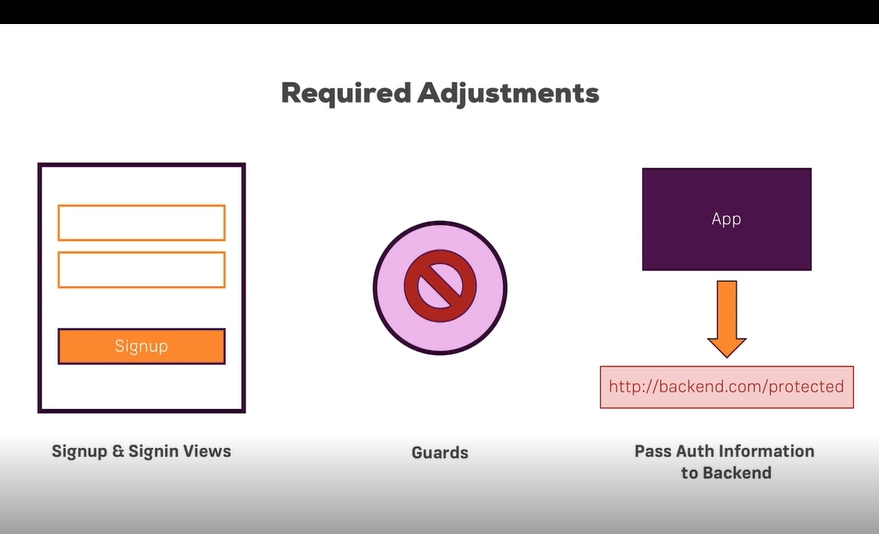
[FORWARDING UNAUTHENTICATED USERS [If user is login then show order else not] 44](#_Toc78124543)

[REDIRECTING THE USER TO THE CHECKOUT PAGE 54](#_Toc78124544)

# UNDERSTANDING AUTHENTICATION IN SINGLE PAGE APPLICATIONS



# REQUIRED ADJUSTMENTS



# ADDING AN AUTH FORM

Auth.js

import React, {Component } from 'react';

import Input from '../../components/UI/Input-Form/Input';

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

import './Auth.css';

class Auth extends Component{

    //state

    state = {

        controls:{

            email: {

                elementType: 'input',

                elementConfig: {

                    type: 'email',

                    placeholder: 'Mail Address'

                },

                value: '',

                validation: {

                    required: true,

                    isEmail: true

                },

                valid: false,

                touched: false

            },

            password: {

                elementType: 'input',

                elementConfig: {

                    type: 'password',

                    placeholder: 'Password'

                },

                value: '',

                validation: {

                    required: true,

                    minLength: 6

                },

                valid: false,

                touched: false

            }

        }

    }

    checkValidty = (value, rules) => {

        let isValid = true;

        //ALTERNATIVE ONE

        if(!rules) {

            return true;

        }

        if(rules.required){

            isValid = value.trim() !== '' && isValid;

        }

        if(rules.minLength){

            isValid = value.length >= rules.minLength && isValid;

        }

        if(rules.isEmail){

            const pattern = /[a-z0-9!#$%&'\*+=/=?]/;

            isValid = pattern.test(value) && isValid;

        }

        if(rules.maxLength){

            isValid = value.length <= rules.maxLength && isValid;

        }

        return isValid;

    }

    render(){

        const formElementsArray = [];

        for(let key in this.state.controls){

            formElementsArray.push({

                id: key,

                config: this.state.controls[key]

            })

        }

        const form = formElementsArray.map(formElement => (

            <Input

                key={formElement.id}

                elementType={formElement.config.elementType}

                elementConfig ={formElement.config.elementConfig}

                value={formElement.config.value}

                invalid={!formElement.config.valid}

                shouldValidate={formElement.config.validation}

                touched={formElement.config.touched}

                changed={(event) => inputChangedHandler(event,formElement.id)}

            />

        ));

        const inputChangedHandler =  (event, controlName) => {

            const updatedControls = {

                ...this.state.controls,

                [controlName]:{

                    ...this.state.controls[controlName],

                    value: event.target.value,

                    valid: this.checkValidty(event.target.value, this.state.controls[controlName].validation),

                    touched: true

                }

            };

            this.setState({controls:updatedControls});

        }

        return(

            <div className='Auth'>

                <form>

                {form}

                <Button btnType="Success">Submit</Button>

                </form>

            </div>

        );

    }

}

export default Auth;

Auth.css

.Auth {

    margin:  20px auto;

    width: 80%;

    text-align: center;

    box-shadow: 0 2px 3px #ccc;

    border: 1px solid #eee;

    padding: 10px;

    box-sizing: border-box;

}

@media (min-width: 600px){

    .Auth {

        width: 500px;

    }

}

# ADDING ACTIONS [submit button action]

Creating actions in store folder named auth.js



In actionTypes.js declaring actions

actionTypes.js

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

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

// export const PURCHASE\_BURGER\_SUCCESS = 'PURCHASE\_BURGER\_SUCCESS';

// export const PURCHASE\_BURGER\_FAIL = 'PURCHASE\_BURGER\_FAIL';

export const AUTH\_START = 'AUTH\_START';

export const AUTH\_SUCCESS = 'AUTH\_SUCCESS';

export const AUTH\_FAIL = 'AUTH\_FAIL';

auth.js[actions]

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

export const authStart = () => {

    return{

        type: actionTypes.AUTH\_START,

//there is not any payload

    };

};

export const authSuccess = (authData) => {

    return{

        type: actionTypes.AUTH\_SUCCESS,

        authData: authData

    };

};

export const authFail = (error) => {

    return{

        type: actionTypes.AUTH\_FAIL,

        error: error

    };

};

//ASYNC CODE for Authentication

export const auth = (email,password) => {

    return dispatch => {

        //authenticate user

        dispatch(authStart());

    }

}

Connecting actions to our container [auth container]

Index.js [actions folder of store folder]

export {

    addIngredient,

    removeIngredient

} from './pizzaBuilder';

export {} from './order';

export {

    auth

} from './auth'

Auth.js [container folder]

import React, {Component } from 'react';

import Input from '../../components/UI/Input-Form/Input';

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

import './Auth.css';

import \* as actions from '../../store/actions/index';

import { connect } from 'react-redux';

class Auth extends Component{

    //state

    state = {

        controls:{

            email: {

                elementType: 'input',

                elementConfig: {

                    type: 'email',

                    placeholder: 'Mail Address'

                },

                value: '',

                validation: {

                    required: true,

                    isEmail: true

                },

                valid: false,

                touched: false

            },

            password: {

                elementType: 'input',

                elementConfig: {

                    type: 'password',

                    placeholder: 'Password'

                },

                value: '',

                validation: {

                    required: true,

                    minLength: 6

                },

                valid: false,

                touched: false

            }

        }

    }

    checkValidty = (value, rules) => {

        let isValid = true;

        //ALTERNATIVE ONE

        if(!rules) {

            return true;

        }

        if(rules.required){

            isValid = value.trim() !== '' && isValid;

        }

        if(rules.minLength){

            isValid = value.length >= rules.minLength && isValid;

        }

        if(rules.isEmail){

            const pattern = /[a-z0-9!#$%&'\*+=/=?]/;

            isValid = pattern.test(value) && isValid;

        }

        if(rules.maxLength){

            isValid = value.length <= rules.maxLength && isValid;

        }

        return isValid;

    }

//4th step of this section

    submitHandler = (event) => {

        event.preventDefault();

        this.props.onAuth(this.state.controls.email.value,this.state.controls.password.value)

    }

    render(){

        const formElementsArray = [];

        for(let key in this.state.controls){

            formElementsArray.push({

                id: key,

                config: this.state.controls[key]

            })

        }

        const form = formElementsArray.map(formElement => (

            <Input

                key={formElement.id}

                elementType={formElement.config.elementType}

                elementConfig ={formElement.config.elementConfig}

                value={formElement.config.value}

                invalid={!formElement.config.valid}

                shouldValidate={formElement.config.validation}

                touched={formElement.config.touched}

                changed={(event) => inputChangedHandler(event,formElement.id)}

            />

        ));

        const inputChangedHandler =  (event, controlName) => {

            const updatedControls = {

                ...this.state.controls,

                [controlName]:{

                    ...this.state.controls[controlName],

                    value: event.target.value,

                    valid: this.checkValidty(event.target.value, this.state.controls[controlName].validation),

                    touched: true

                }

            };

            this.setState({controls:updatedControls});

        }

        return(

            <div className='Auth'>

                <form onSubmit={this.submitHandler}>//3rd step of this section

                {form}

                <Button btnType="Success">Submit</Button>

                </form>

            </div>

        );

    }

}

//1st step of this section[‘adding action’]

const mapDispatchToProps = dispatch => {

    return {

        onAuth: (email,password) => dispatch(actions.auth(email,password))

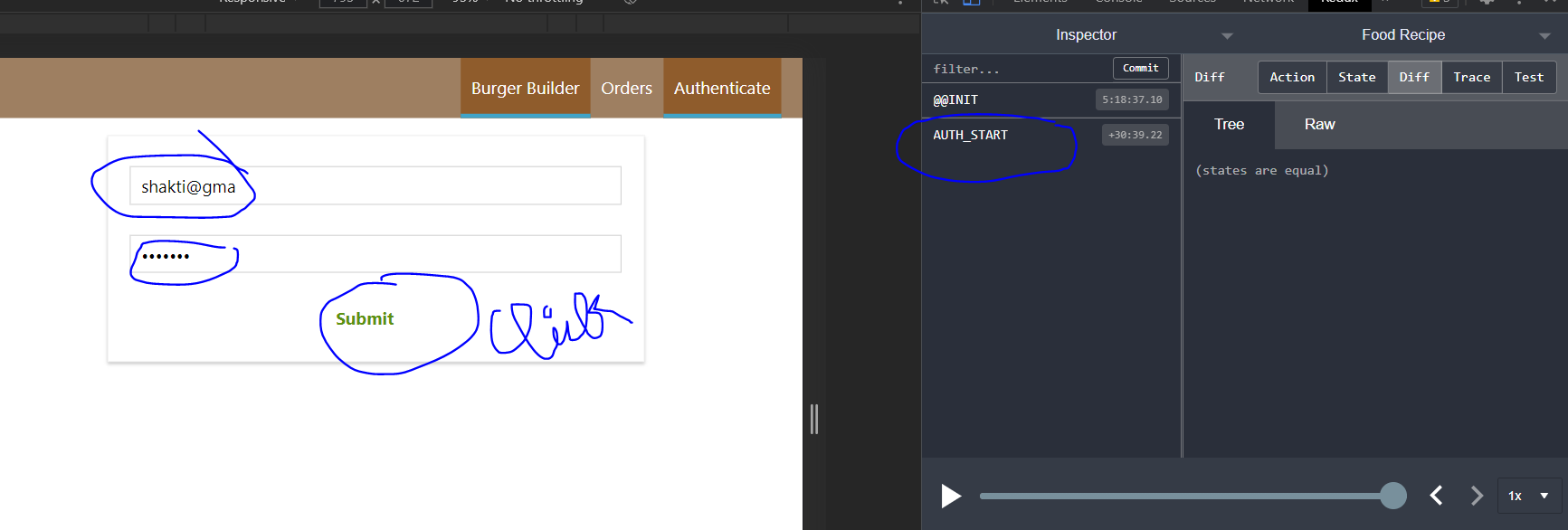
    };

}

//2nd step of this section

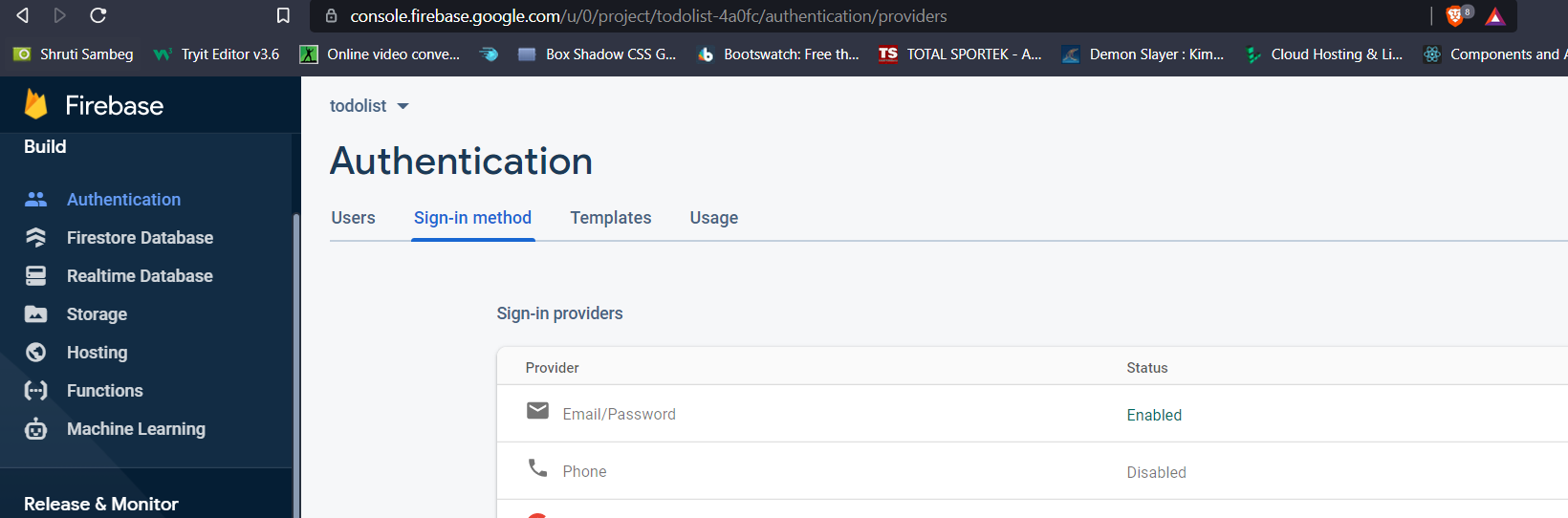
export default connect(null,mapDispatchToProps)(Auth);

On your app[redux devtools]:

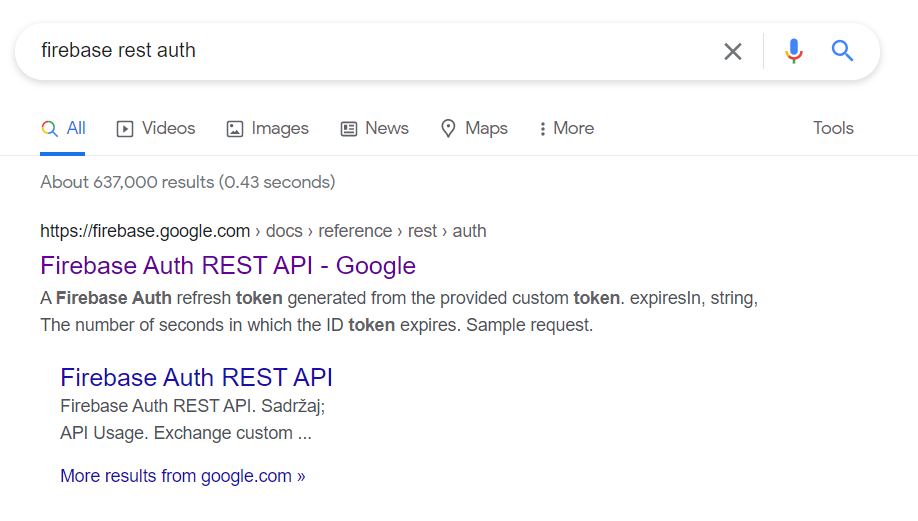


# GETTING A TOKEN FROM THE BACKEND

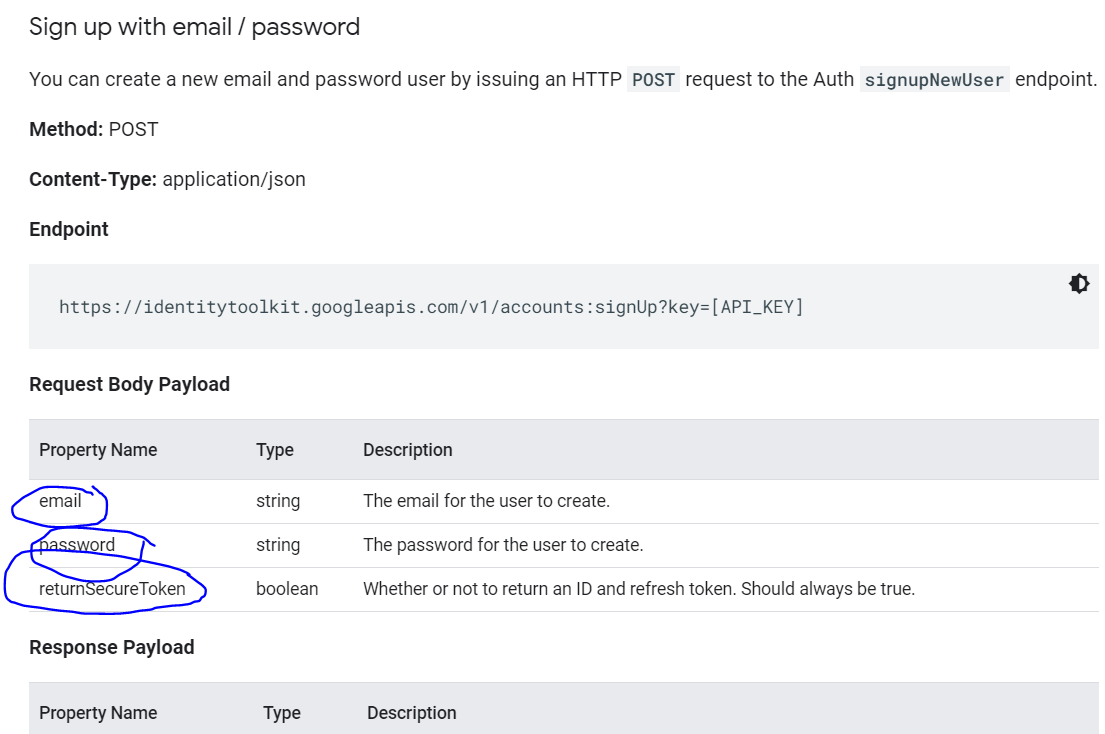
ENABLE FIREBASE AUTHENTICATION



For RESTFUL API, search “ firebase rest auth ” on google



Copy: https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API\_KEY]



auth.js [actions folder]

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

import axios from 'axios';

export const authStart = () => {

    return{

        type: actionTypes.AUTH\_START

    };

};

export const authSuccess = (authData) => {

    return{

        type: actionTypes.AUTH\_SUCCESS,

        authData: authData

    };

};

export const authFail = (error) => {

    return{

        type: actionTypes.AUTH\_FAIL,

        error: error

    };

};

//ASYNC CODE

export const auth = (email,password) => {

    return dispatch => {

        //authenticate

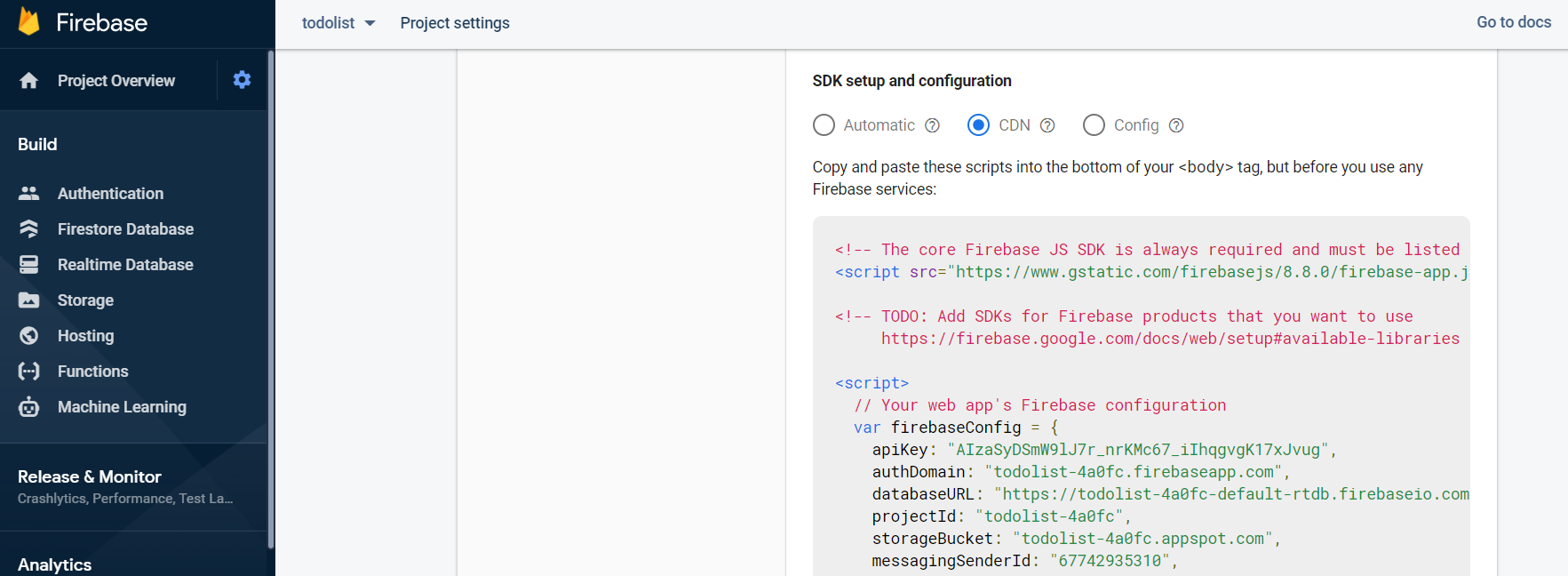
        dispatch(authStart());

        axios.post('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API\_KEY]')

    }

}

REPLACE API\_KEY WITH YOUR FIREBASE\_PROJECT KEY



auth.js [actions folder]

//ASYNC CODE

export const auth = (email,password) => {

    return dispatch => {

        //authenticate

        dispatch(authStart());

        const authData = {

            email: email,

            password: password,

            returnSecureToken: true

        }

        axios.post('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug',authData)

        .then(response => {

            console.log(response);

            dispatch(authSuccess(response.data))

        })

        .catch( err =>{

            console.log(err);

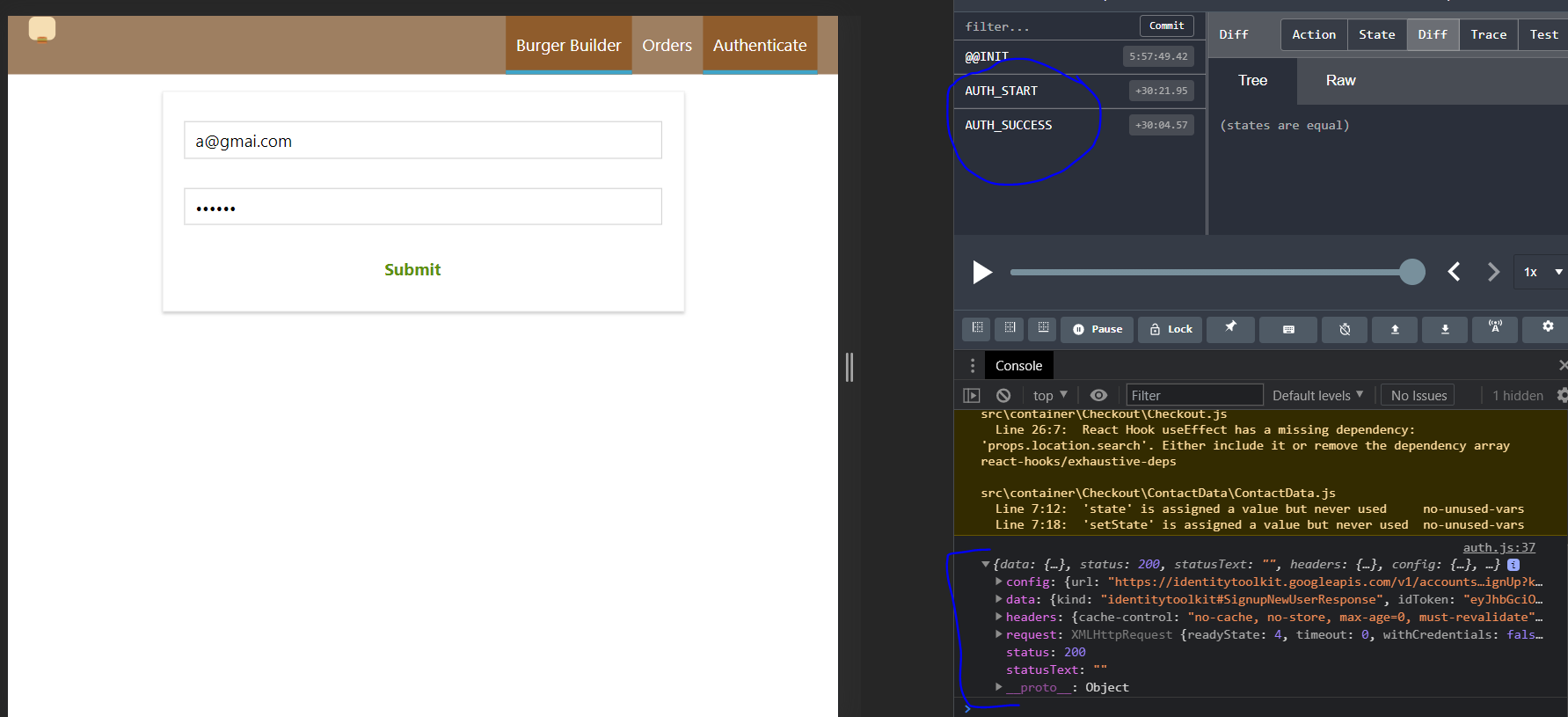
            dispatch(authFail(err))

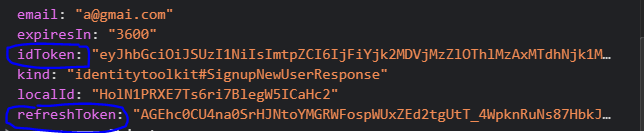
        })

    }

}

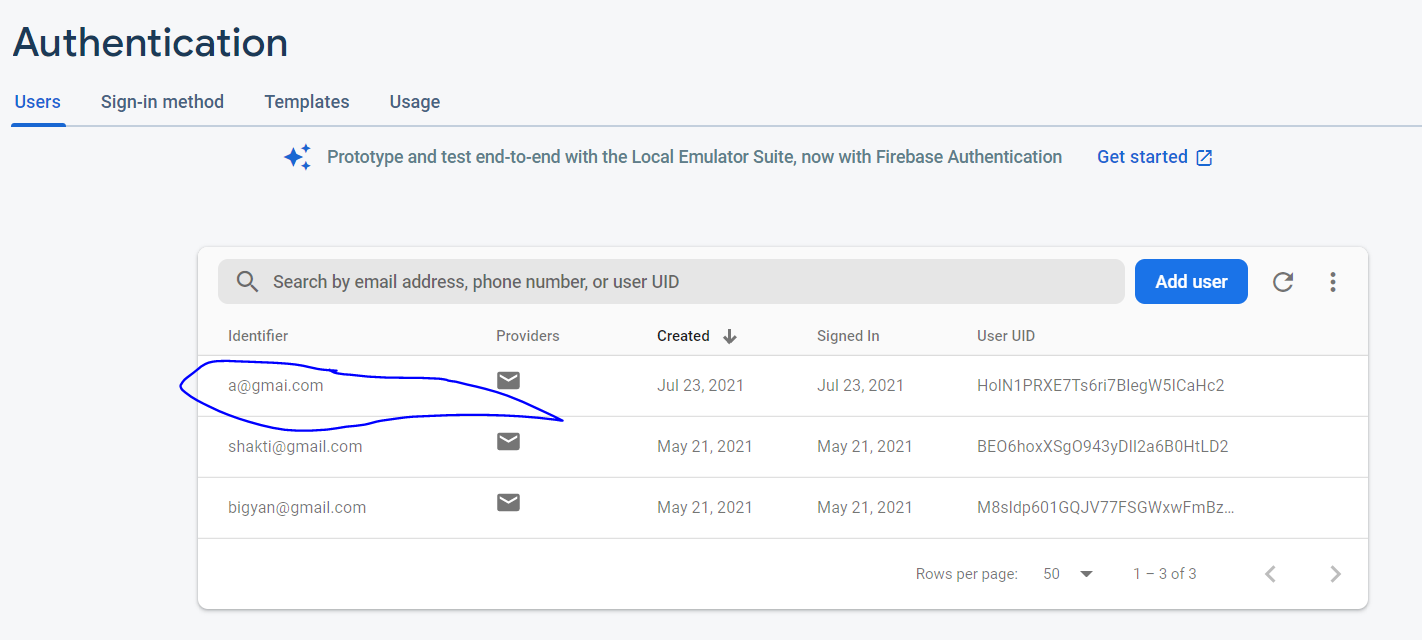
YOUR APP





idToken = user token which expires fast whereas refreshtoken is new generated token

FIREBASE



# ADDING SIGN-IN

Auth.js [container folder]

import React, {Component } from 'react';

import Input from '../../components/UI/Input-Form/Input';

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

import './Auth.css';

import \* as actions from '../../store/actions/index';

import { connect } from 'react-redux';

class Auth extends Component{

    //state

    state = {

        controls:{

            email: {

                elementType: 'input',

                elementConfig: {

                    type: 'email',

                    placeholder: 'Mail Address'

                },

                value: '',

                validation: {

                    required: true,

                    isEmail: true

                },

                valid: false,

                touched: false

            },

            password: {

                elementType: 'input',

                elementConfig: {

                    type: 'password',

                    placeholder: 'Password'

                },

                value: '',

                validation: {

                    required: true,

                    minLength: 6

                },

                valid: false,

                touched: false

            }

        },

        //for sign up

        isSignup: true //2nd step of this section

    }

    checkValidty = (value, rules) => {

        let isValid = true;

        //ALTERNATIVE ONE

        if(!rules) {

            return true;

        }

        if(rules.required){

            isValid = value.trim() !== '' && isValid;

        }

        if(rules.minLength){

            isValid = value.length >= rules.minLength && isValid;

        }

        if(rules.isEmail){

            const pattern = /[a-z0-9!#$%&'\*+=/=?]/;

            isValid = pattern.test(value) && isValid;

        }

        if(rules.maxLength){

            isValid = value.length <= rules.maxLength && isValid;

        }

        return isValid;

    }

    submitHandler = (event) => {

        event.preventDefault();

        this.props.onAuth(this.state.controls.email.value,this.state.controls.password.value,this.state.isSignup)//9th step of this section

    }

//3rd step of this section

    switchAuthModelHandler = () => {

        this.setState(prevState => {

            return{

                isSignup: !prevState.isSignup

            };

        });

    }

    render(){

        const formElementsArray = [];

        for(let key in this.state.controls){

            formElementsArray.push({

                id: key,

                config: this.state.controls[key]

            })

        }

        const form = formElementsArray.map(formElement => (

            <Input

                key={formElement.id}

                elementType={formElement.config.elementType}

                elementConfig ={formElement.config.elementConfig}

                value={formElement.config.value}

                invalid={!formElement.config.valid}

                shouldValidate={formElement.config.validation}

                touched={formElement.config.touched}

                changed={(event) => inputChangedHandler(event,formElement.id)}

            />

        ));

        const inputChangedHandler =  (event, controlName) => {

            const updatedControls = {

                ...this.state.controls,

                [controlName]:{

                    ...this.state.controls[controlName],

                    value: event.target.value,

                    valid: this.checkValidty(event.target.value, this.state.controls[controlName].validation),

                    touched: true

                }

            };

            this.setState({controls:updatedControls});

        }

        return(

            <div className='Auth'>

                <form onSubmit={this.submitHandler}>

                {form}

                <Button btnType="Success">Submit</Button>

                </form>

//1st step of this section

                <Button

                clicked={this.switchAuthModelHandler}

                btnType="Danger"

                >

                SWITCH TO {this.state.isSignup ?

                 'SIGNIN' : 'SIGNUP'}</Button>

            </div>

        );

    }

}

const mapDispatchToProps = dispatch => {

    return {

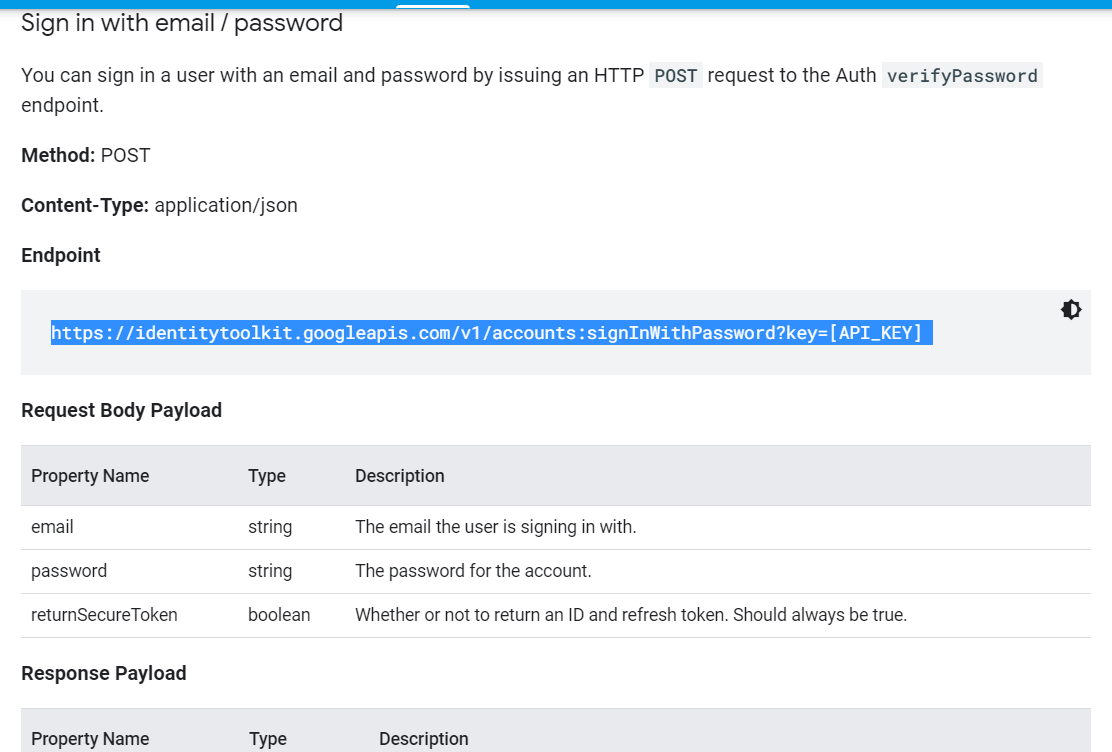
        onAuth: (email,password, isSignup) => dispatch(actions.auth(email,password,isSignup))//8th step of this section

    };

}

export default connect(null,mapDispatchToProps)(Auth);

Signin method of firebase



auth.js

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

import axios from 'axios';

export const authStart = () => {

    return{

        type: actionTypes.AUTH\_START

    };

};

export const authSuccess = (authData) => {

    return{

        type: actionTypes.AUTH\_SUCCESS,

        authData: authData

    };

};

export const authFail = (error) => {

    return{

        type: actionTypes.AUTH\_FAIL,

        error: error

    };

};

//ASYNC CODE

export const auth = (email,password, isSignup) => {//4th step of this section

    return dispatch => {

        //authenticate

        dispatch(authStart());

        const authData = {

            email: email,

            password: password,

            returnSecureToken: true

        };

//5th step of this section

        let url = 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug';

        if(!isSignup){

            url='https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug';

        }

        axios.post(url,authData)//7th step

        .then(response => {

            console.log(response);

            dispatch(authSuccess(response.data))

        })

        .catch( err =>{

            console.log(err);

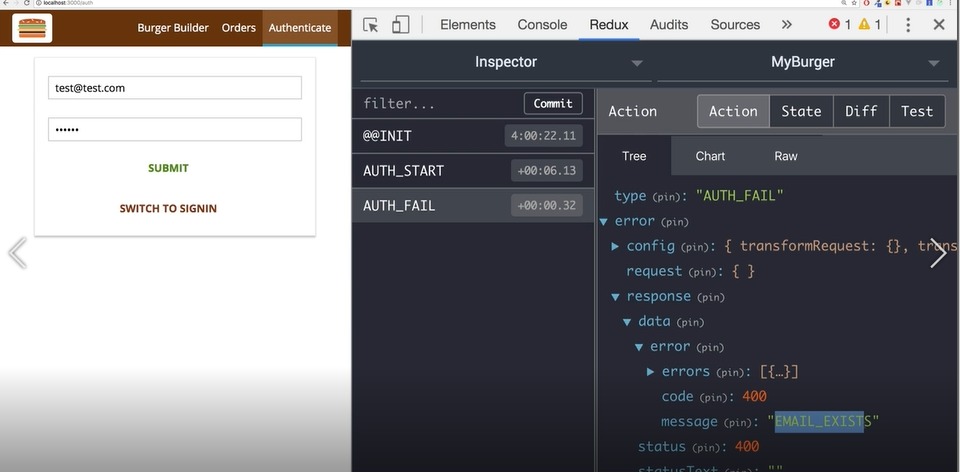
            dispatch(authFail(err))

        })

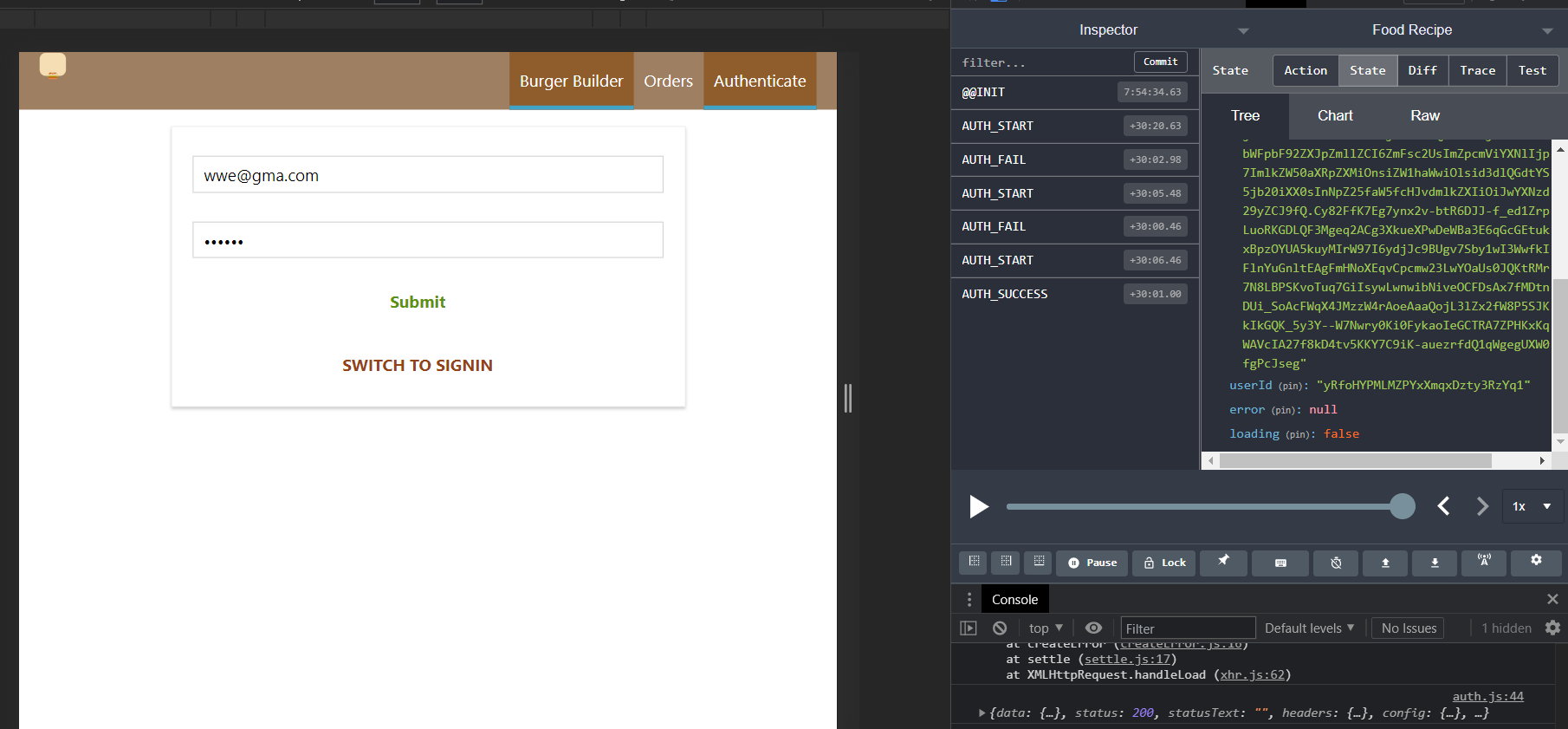
    }

}

Your APP



# STORING THE TOKEN



Create new reducer file in reducer named: auth.js

auth.js

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

import { updateObject } from '../utility';

const initialState ={

    token: null,

    userId: null,

    error: null,

    loading: false

}

const authStart = (state,action) =>{

    return updateObject(state,{error: null,loading: true})

}

const authSuccess = (state,action) =>{

    return updateObject(state,{

        token: action.idToken,

        userId: action.userId,

        error: null,

        loading: false

    })

}

const authFail = (state,action) =>{

    return updateObject(state,{

        error: action.error,

        loading: false

    })

}

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

    switch ((action.type)) {

        case actionTypes.AUTH\_START:

            return authStart(state,action);

        case actionTypes.AUTH\_SUCCESS:

            return authSuccess(state,action);

        case actionTypes.AUTH\_FAIL:

            return authFail(state,action);

        default:

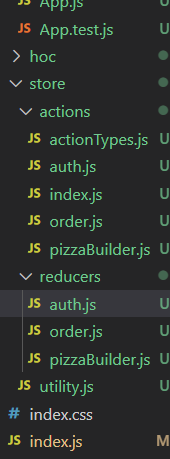
            return state;

    }

};

export default reducer;

Create utility.js inside store folder:



utility.js

export const updateObject = (oldObject, updatedProperties) => {

    return{

        ...oldObject,

        ...updatedProperties

    }

}

#Now combining two reducers in index.js

Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './container/App';

import reportWebVitals from './reportWebVitals';

import thunk from 'redux-thunk';

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

import { Provider } from 'react-redux';

import {createStore,applyMiddleware,compose,combineReducers} from 'redux';

import burgerBuilderReducer from './store/reducers/pizzaBuilder';

import authReducer from './store/reducers/auth';

const rootReducer = combineReducers({

  burgerBuilder: burgerBuilderReducer,

  auth: authReducer

});

const composeEnhancers = window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;

const store = createStore(rootReducer,composeEnhancers(

  applyMiddleware(thunk)

));

const app = (

  <Provider store={store}>

    <BrowserRouter>

      <App />

    </BrowserRouter>

  </Provider>

);

ReactDOM.render(app,document.getElementById('root')

);

// 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();

PizzaBuilder.js [container folder]

const mapStateToProps = state =>{

    return{

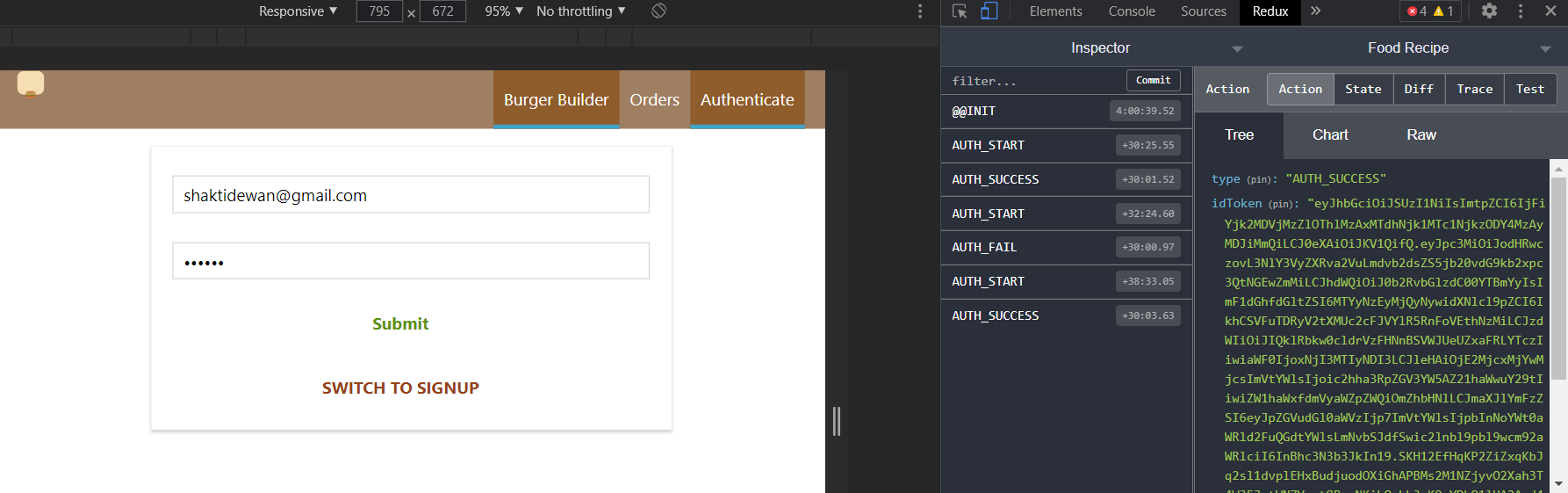
        ings: state.burgerBuilder.ingredients,

        price: state.burgerBuilder.totalPrice

    };

}

Application



Auth.js[actions folder]

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

import axios from 'axios';

export const authStart = () => {

    return{

        type: actionTypes.AUTH\_START

    };

};

export const authSuccess = (token, userId) => {

    return{

        type: actionTypes.AUTH\_SUCCESS,

        idToken: token,

        userId: userId

    };

};

export const authFail = (error) => {

    return{

        type: actionTypes.AUTH\_FAIL,

        error: error

    };

};

//ASYNC CODE

export const auth = (email,password, isSignup) => {

    return dispatch => {

        //authenticate

        dispatch(authStart());

        const authData = {

            email: email,

            password: password,

            returnSecureToken: true

        };

        let url = 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        if(!isSignup){

            url='https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        }

        axios.post(url,authData)

        .then(response => {

            console.log(response);

            dispatch(authSuccess(response.data.idToken, response.data.localId))

        })

        .catch( err =>{

            console.log(err);

            dispatch(authFail(err))

        })

    }

}

# Logging users out

FLOW: auth.js[actions] -> actionTypes[for action type const] -> auth.js[actions for dispatching] -> auth.js[reducer for handing logout case]

Auth.js[actions folder]

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

import axios from 'axios';

export const authStart = () => {

    return{

        type: actionTypes.AUTH\_START

    };

};

export const authSuccess = (token, userId) => {

    return{

        type: actionTypes.AUTH\_SUCCESS,

        idToken: token,

        userId: userId

    };

};

export const authFail = (error) => {

    return{

        type: actionTypes.AUTH\_FAIL,

        error: error

    };

};

//for logout

export const logout = () => {

    //sync

    return{

        type: actionTypes.AUTH\_LOGOUT//6th step; dispatching

    }

}//3rd step of this section

//1st step of this section

export const checkAuthTimeout = (expirationTIme) => {

    return dispatch =>{

        setTimeout(() => {

            //async

            dispatch(logout());//4th step

        },expirationTIme);

    // },expirationTIme\*1000); //for one hour

    };

};

//ASYNC CODE

export const auth = (email,password, isSignup) => {

    return dispatch => {

        //authenticate

        dispatch(authStart());

        const authData = {

            email: email,

            password: password,

            returnSecureToken: true

        };

        let url = 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        if(!isSignup){

            url='https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        }

        axios.post(url,authData)

        .then(response => {

            console.log(response);

            dispatch(authSuccess(response.data.idToken, response.data.localId))

            dispatch(checkAuthTimeout(response.data.expiresIn));//2nd step of this section, expiresIn can be seen in console

        })

        .catch( err =>{

            console.log(err);

            dispatch(authFail(err))

        })

    }

}

actionTypes:

//for log out

export const AUTH\_LOGOUT = 'AUTH\_LOGOUT';//5th step

auth.js[reducer]

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

import { updateObject } from '../utility';

const initialState ={

    token: null,

    userId: null,

    error: null,

    loading: false

}

const authStart = (state,action) =>{

    return updateObject(state,{error: null,loading: true})

}

const authSuccess = (state,action) =>{

    return updateObject(state,{

        token: action.idToken,

        userId: action.userId,

        error: null,

        loading: false

    })

}

const authFail = (state,action) =>{

    return updateObject(state,{

        error: action.error,

        loading: false

    })

}

//7th step

const authLogout = (state, action) => {

    return updateObject(state, {token: null, userId: null});

}

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

    switch ((action.type)) {

        case actionTypes.AUTH\_START:

            return authStart(state,action);

        case actionTypes.AUTH\_SUCCESS:

            return authSuccess(state,action);

        case actionTypes.AUTH\_FAIL:

            return authFail(state,action);

        case actionTypes.AUTH\_LOGOUT:

            return authLogout(state,action);//8th step

        default:

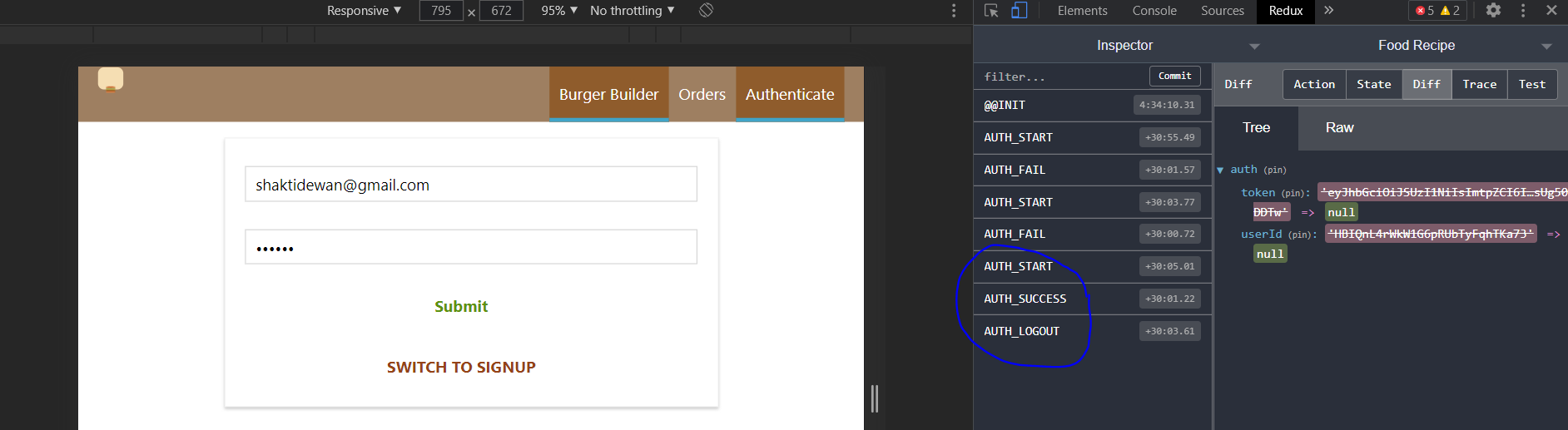
            return state;

    }

};

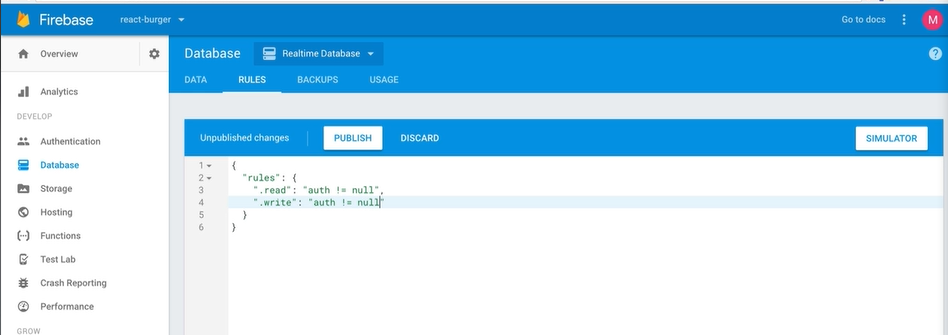
export default reducer;

APP:

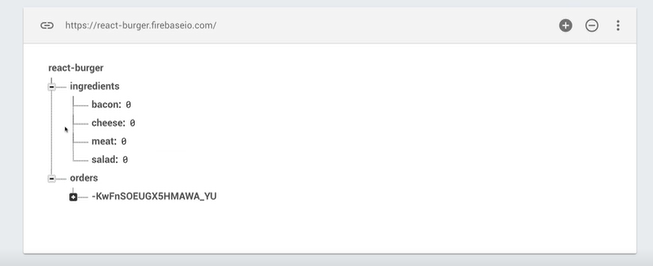


# ACCESSING PROTECTED RESOURCES [access to data based on authentication]

In firebase,

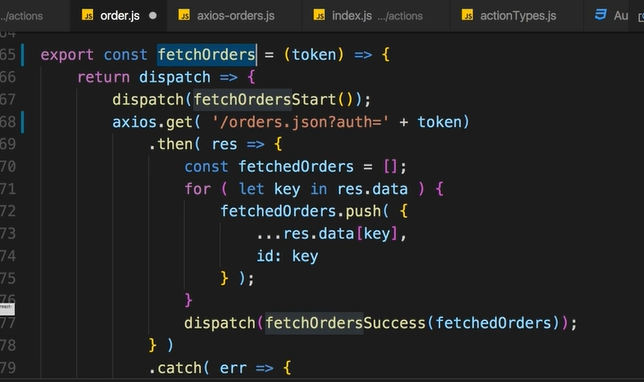


But we will give access to ingrdients not to orders.

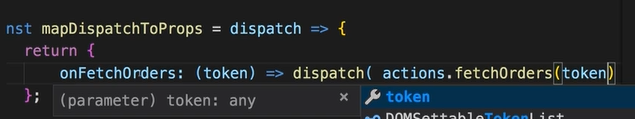


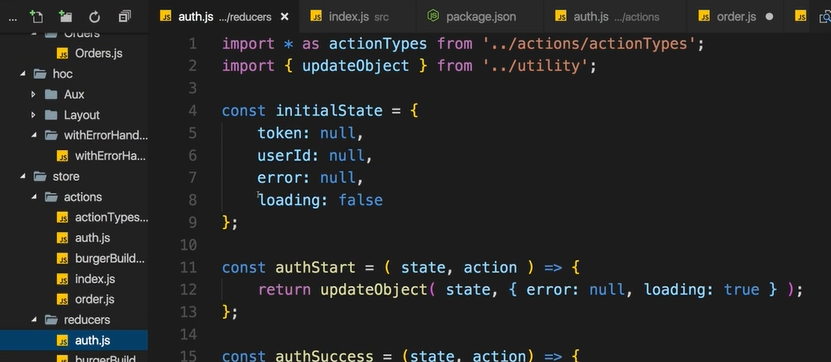


Order.js [actions folder]

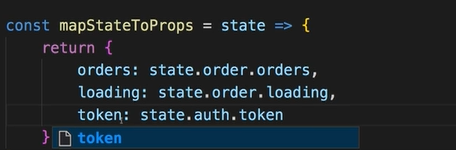


Orders.js [container folder]



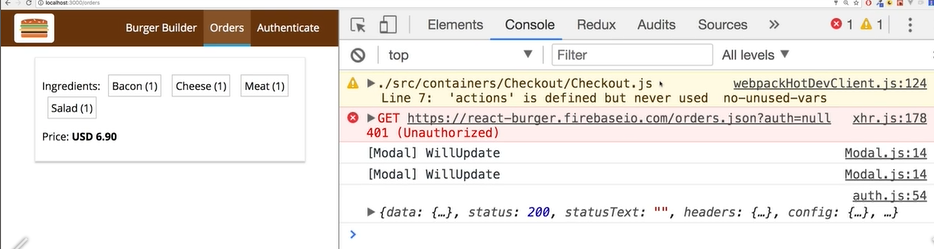
Token is stored in auth.js reducer

Orders.js [container folder]

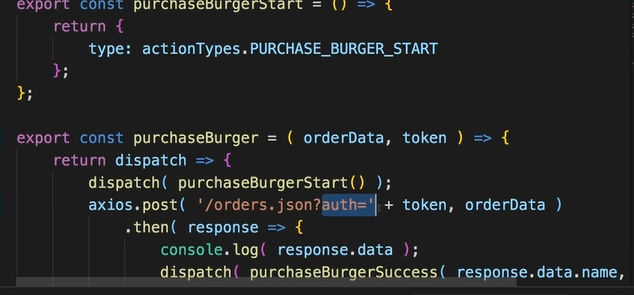




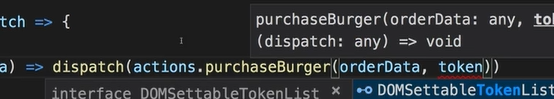
APP:

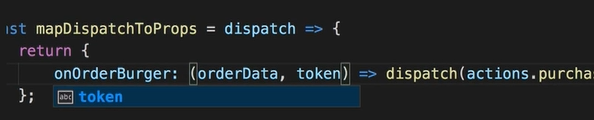


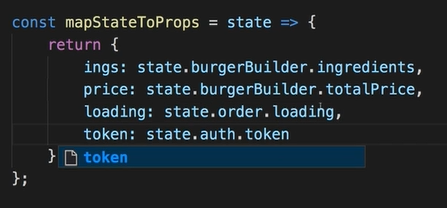
FOR MAKING ORDER:

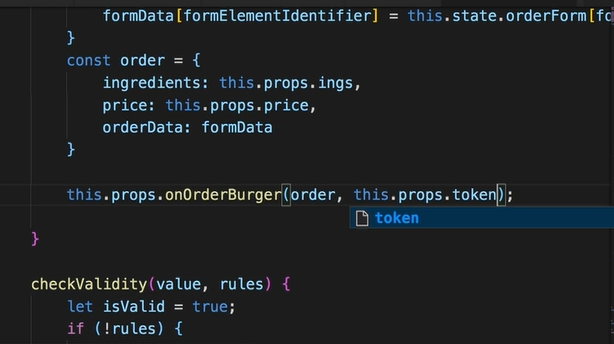
Order.js [reducer] 

contactData.js[container folder]

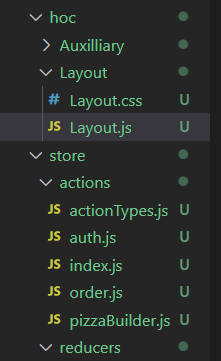




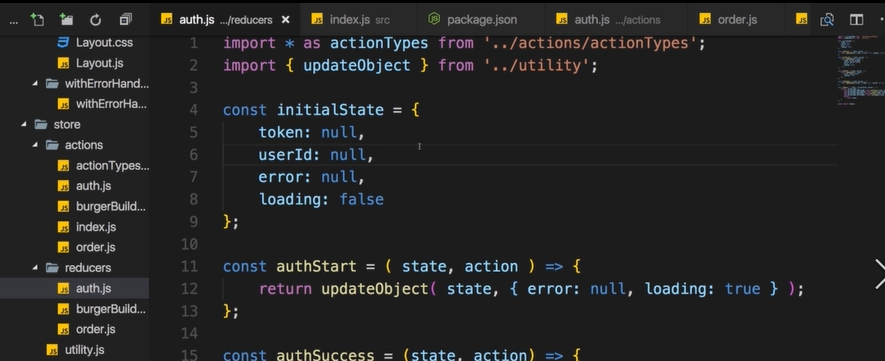




# UPDATING THE UI DEPENDING ON UI STATE [showing logout in navigation]:



REFERENCE to find authenticate user on basis of token is null or not:



Layout.js

import React,{useState} from 'react';

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

import Toolbar from '../../components/Navigation/Toolbar/Toolbar';

import  './Layout.css';

import SideDrawer from '../../components/Navigation/SideBar/SideDrawer';

import { connect } from 'react-redux';//1st

const Layout = (props) => {

    const [state, setstate] = useState({

        showSideDrawer: true,

    });

    const SideDrawerCLosedHandler = () => {

        setstate({showSideDrawer: false})

    }

    const sideDrawerToggleClicked = () => {

        // setstate({

        //     showSideDrawer: !state.showSideDrawer

        // })

        // THIS WILL THROW AN ERROR

        setstate((prevState) => {

            return {showSideDrawer: !prevState.showSideDrawer};

        })

    }

    return(

        <Aux>

            <Toolbar

            isAuth = {props.isAuthenticated}//4th

            drawerToggleClicked={sideDrawerToggleClicked} />

            <SideDrawer

            isAuth = {props.isAuthenticated}//5th

            open={state.showSideDrawer} closed={SideDrawerCLosedHandler} />

            <main className='Content'>

                {props.children}

            </main>

        </Aux>

    );

}

//2nd

const mapStateToProps = state => {

    return{

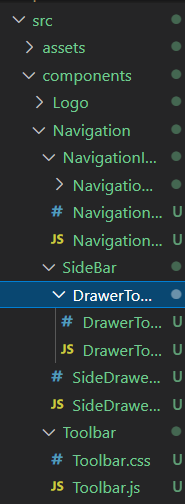
        //JS object

        isAuthenticated: state.auth.token !== null // auth is reducer store name

    };

};

export default connect(mapStateToProps)(Layout);//3rd



Toolbar.js

import React from 'react';

import './Toolbar.css';

import Logo  from '../../Logo/Logo';

import NavigationItems from '../NavigationItems/NavigationItems';

import DrawerToggle from '../SideBar/DrawerToggle/DrawerToggle';

const Toolbar = (props) => (

    <header className='Toolbar'>

        <DrawerToggle clicked={props.drawerToggleClicked}/>

        <Logo/>

        <nav className='DesktopOnly '>

            <NavigationItems isAuthenticated={props.isAuth}/> //6th

        </nav>

    </header>

);

export default Toolbar;

SideDrawer.js

import React from 'react';

import './SideDrawer.css'

import Logo from '../../Logo/Logo';

import NavigationItems from '../NavigationItems/NavigationItems';

import Backdrop from '../../UI/Backdrop/Backdrop';

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

const SideDrawer = (props) => {

    let attachedClasses = ['SideDrawer','Close'];

    if(props.open) {

        attachedClasses = ['SideDrawer','Open'];

    }

    return (

        <Aux>

        <Backdrop show={props.open} clicked={props.closed} />

        <div className={attachedClasses.join(' ')}>

            <Logo />

             <nav>

                <NavigationItems isAuthenticated={props.isAuth}/>//7th

            </nav>

        </div>

        </Aux>

        );

}

export default SideDrawer;

IF needed, DrawerToggle

import React from 'react';

import './DrawerToggle.css';

const DrawerToggle = (props) => (

    <div onClick={props.clicked} className='DrawerToggle'>

        <div></div>

        <div></div>

        <div></div>

    </div>

);

export default DrawerToggle;

NavigationsItems.js

import React from 'react';

import './NavigationItems.css';

import NavigationItem from './NavigationItem/NavigationItem';

const NavigationItems = (props) => (

    <ul className='NavigationItems'>

        <NavigationItem link="/" extact>

        {/\* //active is boolean so it can be passed like this \*/}

            Burger Builder

        </NavigationItem>

        <NavigationItem link="/orders">

            Orders

        </NavigationItem>

//8th

        { !props.isAuthenticated

            ? <NavigationItem link="/auth">

                Authenticate

            </NavigationItem>

            : <NavigationItem link="/logout">

                Logout

            </NavigationItem>

        }

    </ul>

);

export default NavigationItems;

NavigationItem.js

import React from 'react';

import './NavigationItem.css';

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

const NavigationItem = (props) => (

    <li className='NavigationItem'>

    <NavLink

        to={props.link}

        activeClassName='active'

        exact={props.exact}

    >

    {props.children}

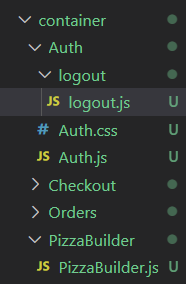
    </NavLink>

    </li>

);

export default NavigationItem;

# ADDING LOGOUT LINK [logout and redirect to home page]



Create Logout folder.

Reference: we have reducer logout:

//for logout

export const logout = () => {

    //sync

    return{

        type: actionTypes.AUTH\_LOGOUT

    }

}

export const checkAuthTimeout = (expirationTIme) => {

    return dispatch =>{

        setTimeout(() => {

            //async

            dispatch(logout());

        // },expirationTIme);

    },expirationTIme\*1000); //for one hour

    };

};

Logout.js

import React, {Component} from 'react';

import { Redirect } from 'react-router-dom';

import {connect } from 'react-redux';

import \* as actions from '../../../store/actions/index';

class Logout extends Component {

    componentDidMount(){

        this.props.onLogout();

    }

    render () {

        return <Redirect to="/" />;

    }

}

const mapDispatchToProps = dispatch =>{

    return {

        onLogout: () => dispatch(actions.logout())

    };

}

export default connect(null,mapDispatchToProps)(Logout);

App.js

import React from 'react';

import Layout from '../hoc/Layout/Layout';

import PizzaBuilder  from './PizzaBuilder/PizzaBuilder';

import Checkout from './Checkout/Checkout';

import Orders from './Orders/Orders';

import { Route,Switch } from 'react-router-dom';

import Auth from './Auth/Auth';

import Logout from './Auth/logout/logout';

const App = () => {

  return(

    <div className="App">

      <Layout>

        <Switch>

          <Route path="/checkout" component={Checkout} />

          <Route path="/orders" component={Orders} />

          <Route path="/auth" component={Auth} />

          <Route path="/logout" component={Logout} />

          <Route path="/" exact component={PizzaBuilder} />

        </Switch>

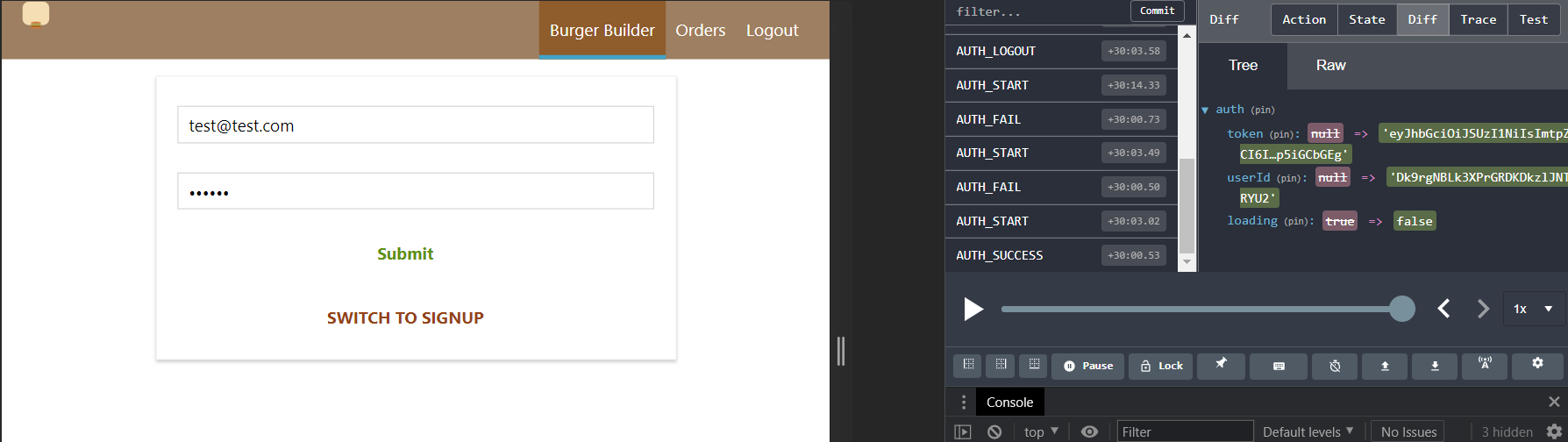
      </Layout>

    </div>

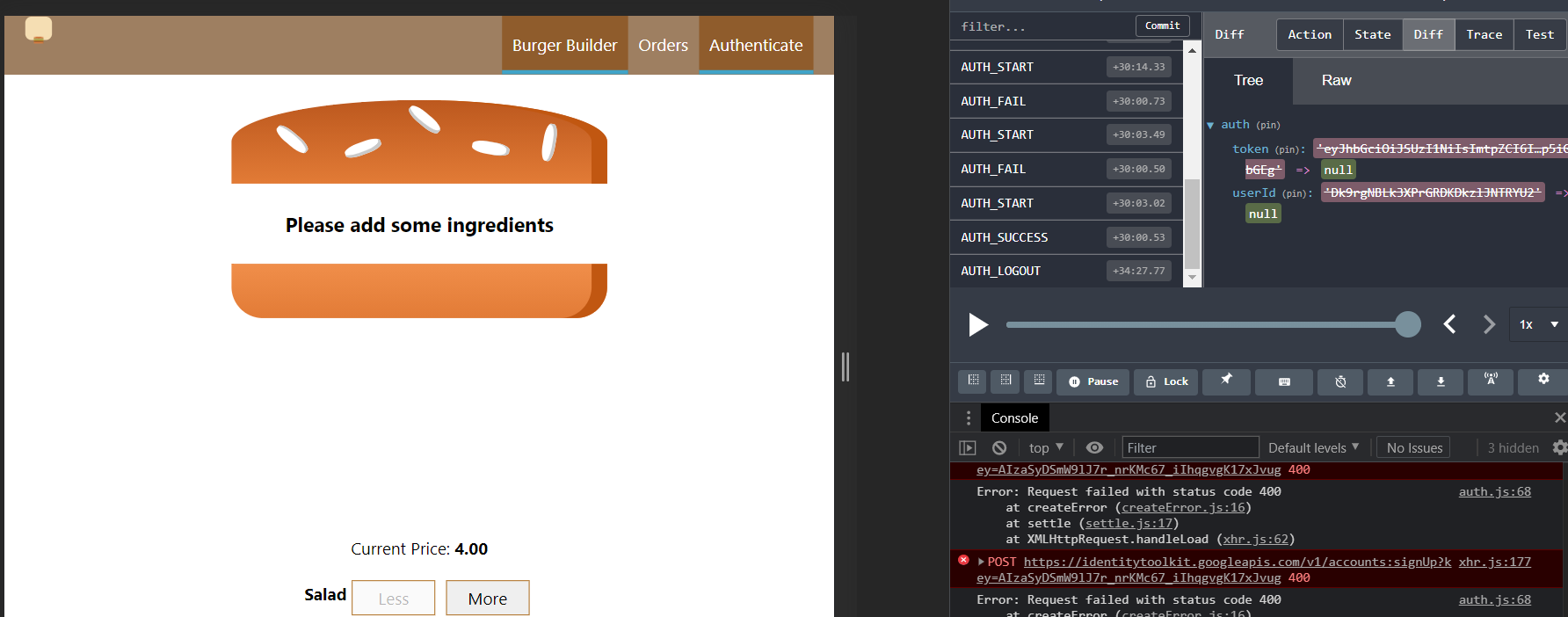
  )

}

export default App;

APP:

After logout click:



# FORWARDING UNAUTHENTICATED USERS [If user is login then show order else not]

Navigation.js

import React from 'react';

import './NavigationItems.css';

import NavigationItem from './NavigationItem/NavigationItem';

const NavigationItems = (props) => (

    <ul className='NavigationItems'>

        <NavigationItem link="/" extact>

        {/\* //active is boolean so it can be passed like this \*/}

            Burger Builder

        </NavigationItem>

        {props.isAuthenticated

            ?<NavigationItem link="/orders">

                Orders

            </NavigationItem>

            : null

        }

        { !props.isAuthenticated

            ? <NavigationItem link="/auth">

                Authenticate

            </NavigationItem>

            : <NavigationItem link="/logout">

                Logout

            </NavigationItem>

        }

    </ul>

);

export default NavigationItems;

**REDIRECTING TO HOME PAGE AFTER SUBMIT BUTTOM**

**Auth.js[container folder]**

import React, {Component } from 'react';

import Input from '../../components/UI/Input-Form/Input';

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

import './Auth.css';

import \* as actions from '../../store/actions/index';

import { connect } from 'react-redux';

import {Redirect} from 'react-router-dom';

class Auth extends Component{

    //state

    state = {

        controls:{

            email: {

                elementType: 'input',

                elementConfig: {

                    type: 'email',

                    placeholder: 'Mail Address'

                },

                value: '',

                validation: {

                    required: true,

                    isEmail: true

                },

                valid: false,

                touched: false

            },

            password: {

                elementType: 'input',

                elementConfig: {

                    type: 'password',

                    placeholder: 'Password'

                },

                value: '',

                validation: {

                    required: true,

                    minLength: 6

                },

                valid: false,

                touched: false

            }

        },

        //for sign up

        isSignup: true

    }

    checkValidty = (value, rules) => {

        let isValid = true;

        //ALTERNATIVE ONE

        if(!rules) {

            return true;

        }

        if(rules.required){

            isValid = value.trim() !== '' && isValid;

        }

        if(rules.minLength){

            isValid = value.length >= rules.minLength && isValid;

        }

        if(rules.isEmail){

            const pattern = /[a-z0-9!#$%&'\*+=/=?]/;

            isValid = pattern.test(value) && isValid;

        }

        if(rules.maxLength){

            isValid = value.length <= rules.maxLength && isValid;

        }

        return isValid;

    }

    submitHandler = (event) => {

        event.preventDefault();

        this.props.onAuth(this.state.controls.email.value,this.state.controls.password.value,this.state.isSignup)

    }

    switchAuthModelHandler = () => {

        this.setState(prevState => {

            return{

                isSignup: !prevState.isSignup

            };

        });

    }

    render(){

        const formElementsArray = [];

        for(let key in this.state.controls){

            formElementsArray.push({

                id: key,

                config: this.state.controls[key]

            })

        }

        const form = formElementsArray.map(formElement => (

            <Input

                key={formElement.id}

                elementType={formElement.config.elementType}

                elementConfig ={formElement.config.elementConfig}

                value={formElement.config.value}

                invalid={!formElement.config.valid}

                shouldValidate={formElement.config.validation}

                touched={formElement.config.touched}

                changed={(event) => inputChangedHandler(event,formElement.id)}

            />

        ));

        const inputChangedHandler =  (event, controlName) => {

            const updatedControls = {

                ...this.state.controls,

                [controlName]:{

                    ...this.state.controls[controlName],

                    value: event.target.value,

                    valid: this.checkValidty(event.target.value, this.state.controls[controlName].validation),

                    touched: true

                }

            };

            this.setState({controls:updatedControls});

        }

        //redirect to

        let authRedirect = null;

        if(this.props.isAuthenticated){

            authRedirect =<Redirect to="/" />

        }

        return(

            <div className='Auth'>

                {authRedirect}

                <form onSubmit={this.submitHandler}>

                {form}

                <Button btnType="Success">Submit</Button>

                </form>

                <Button

                clicked={this.switchAuthModelHandler}

                btnType="Danger"

                >

                SWITCH TO {this.state.isSignup ?

                 'LOGIN' : 'SIGNUP'}</Button>

            </div>

        );

    }

}

const mapStateToProps = state =>{

    return {

        isAuthenticated: state.auth.token !== null

    };

};

const mapDispatchToProps = dispatch => {

    return {

        onAuth: (email,password, isSignup) => dispatch(actions.auth(email,password,isSignup))

    };

}

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

**Need to sign up/ log in to make burger order.**



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';

import \* as pizzaBuilderActions from '../../store/actions/index'

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 ;

    }

    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 = {updatePurchaseState(props.ings)}

                price={props.price}

                isAuth = {props.isAuthenticated}

             />

        </Aux>

    );

}

const mapStateToProps = state =>{

    return{

        ings: state.burger.ingredients,

        price: state.burger.totalPrice,

        isAuthenticated: state.auth.token !== null

    };

}

const mapDispatchToProps = dispatch => {

    return{

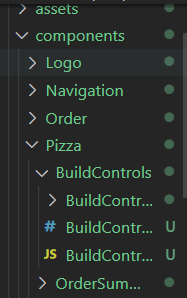
        onIngredientAdded: (ingName) => dispatch(pizzaBuilderActions.addIngredient(ingName)),

        onIngredien tRemoved: (ingName) => dispatch(pizzaBuilderActions.removeIngredient(ingName))

    }

}

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



BuildCOntrol.js

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}

         >{props.isAuth ? 'ORDER NOW' : 'SIGN UP TO MAKE ORDER'}</button>

    </div>

);

export default BuildControls;

TO PUSH USER TO AUTH PAGE WHEN MAKING ORDER IF NOT LOGIN

PIzzaBuilder.js

const purchaseHandler = () => {

        if(props.isAuthenticated){

            setpurchases({purchasing: true});

        }

        else {

            props.history.push('/auth');

        }

    }

# REDIRECTING THE USER TO THE CHECKOUT PAGE

Pizzabuilder.js [reducer folder]

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

const initialState= {

    ingredients: {

        salad: 0,

        bacon: 0,

        cheese: 0,

        meat: 0

    },

    totalPrice: 4,

    buidling: false

}

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{

                //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

                },

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

                buidling: true

            };

        case actionTypes.REMOVE\_INGREDIENTS:

            return{

                //copying old state

                ...state,

                ingredients: {

                    ...state.ingredients,

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

                },

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

                buidling: false

            };

        default:

            return state;

    }

}

export default Reducer;

Auth.js [reducer folder]

const initialState ={

    token: null,

    userId: null,

    error: null,

    loading: false,

    authRedirectPath: '/'

}

actionTypes.js [actions folder]

export const SET\_AUTH\_REDIRECT\_PATH = 'SET\_AUTH\_REDIRECT\_PATH'

auth.js [actions folder]

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

import axios from 'axios';

export const authStart = () => {

    return{

 …………………………………

//ASYNC CODE

export const auth = (email,password, isSignup) => {

………………………………………..

}

export const setAuthRedirectPath = (path) => {

    return {

        type: actionTypes.SET\_AUTH\_REDIRECT\_PATH,

        //payload

        path: path

    }

}

**HANDLE this action in REDUCERS**

Auth.js[reducer folder]

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

import { updateObject } from '../utility';

const initialState ={

    token: null,

    userId: null,

    error: null,

    loading: false,

    authRedirectPath: '/'

}

const authStart = (state,action) =>{

    ………………… . . . . . . . . .

const setAuthRedirectPath = (state, action) => {

    return updateObject(state, {authRedirectPath: action.path})

}

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

    switch ((action.type)) {

        case actionTypes.AUTH\_START:

            return authStart(state,action);

        case actionTypes.AUTH\_SUCCESS:

            return authSuccess(state,action);

        case actionTypes.AUTH\_FAIL:

            return authFail(state,action);

        case actionTypes.AUTH\_LOGOUT:

            return authLogout(state,action);

        case actionTypes.SET\_AUTH\_REDIRECT\_PATH:

            return setAuthRedirectPath(state,action);

        default:

            return state;

    }

};

export default reducer;

**DISPATCH THE ACTION FUNCTION**

**ADDING TO INDEX TO MAKE AVAILABLE FOR OTHERS FILES**

Index.js [actions folder]

export {

    auth,

    logout,

    setAuthRedirectPath

} from './auth'

PizzaBuilder.js [ container folder]

import \* as pizzaBuilderActions from '../../store/actions/index'

const PizzaBuilder = (props) => {

    const [purchases, setpurchases] = useState({

        purchasing: false

    })

    const purchaseHandler = () => {

        if(props.isAuthenticated){

            setpurchases({purchasing: true});

        }

        else {

            props.onSetAuthRedirectPath('/checkout');

            props.history.push('/auth');

        }

    }

const mapDispatchToProps = dispatch => {

    return{

        onIngredientAdded: (ingName) => dispatch(pizzaBuilderActions.addIngredient(ingName)),

        onIngredientRemoved: (ingName) => dispatch(pizzaBuilderActions.removeIngredient(ingName)),

        onSetAuthRedirectPath: (path) => dispatch(pizzaBuilderActions.setAuthRedirectPath(path))

    }

}

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

Auth.js [container folder]

import React, {Component } from 'react';

import Input from '../../components/UI/Input-Form/Input';

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

import './Auth.css';

import \* as actions from '../../store/actions/index';

import { connect } from 'react-redux';

import {Redirect} from 'react-router-dom';

class Auth extends Component{

    //state

    state = {

        controls:{

            email: {

                elementType: 'input',

                elementConfig: {

                    type: 'email',

                    placeholder: 'Mail Address'

                },

                value: '',

                validation: {

                    required: true,

                    isEmail: true

                },

                valid: false,

                touched: false

            },

            password: {

                elementType: 'input',

                elementConfig: {

                    type: 'password',

                    placeholder: 'Password'

                },

                value: '',

                validation: {

                    required: true,

                    minLength: 6

                },

                valid: false,

                touched: false

            }

        },

        //for sign up

        isSignup: true

    }

    componentDidMount(){

        if(!this.props.buildingBurger && this.props.authRedirectPath !== '/' ){

            this.props.onSetAuthRedirectPath();

        }

    }

    checkValidty = (value, rules) => {

        let isValid = true;

        //ALTERNATIVE ONE

        if(!rules) {

            return true;

        }

        if(rules.required){

            isValid = value.trim() !== '' && isValid;

        }

        if(rules.minLength){

            isValid = value.length >= rules.minLength && isValid;

        }

        if(rules.isEmail){

            const pattern = /[a-z0-9!#$%&'\*+=/=?]/;

            isValid = pattern.test(value) && isValid;

        }

        if(rules.maxLength){

            isValid = value.length <= rules.maxLength && isValid;

        }

        return isValid;

    }

    submitHandler = (event) => {

        event.preventDefault();

        this.props.onAuth(this.state.controls.email.value,this.state.controls.password.value,this.state.isSignup)

    }

    switchAuthModelHandler = () => {

        this.setState(prevState => {

            return{

                isSignup: !prevState.isSignup

            };

        });

    }

    render(){

        const formElementsArray = [];

        for(let key in this.state.controls){

            formElementsArray.push({

                id: key,

                config: this.state.controls[key]

            })

        }

        const form = formElementsArray.map(formElement => (

            <Input

                key={formElement.id}

                elementType={formElement.config.elementType}

                elementConfig ={formElement.config.elementConfig}

                value={formElement.config.value}

                invalid={!formElement.config.valid}

                shouldValidate={formElement.config.validation}

                touched={formElement.config.touched}

                changed={(event) => inputChangedHandler(event,formElement.id)}

            />

        ));

        const inputChangedHandler =  (event, controlName) => {

            const updatedControls = {

                ...this.state.controls,

                [controlName]:{

                    ...this.state.controls[controlName],

                    value: event.target.value,

                    valid: this.checkValidty(event.target.value, this.state.controls[controlName].validation),

                    touched: true

                }

            };

            this.setState({controls:updatedControls});

        }

        //redirect to

        let authRedirect = null;

        if(this.props.isAuthenticated){

            authRedirect =<Redirect to={this.props.authRedirectPath} />

        }

        return(

            <div className='Auth'>

                {authRedirect}

                <form onSubmit={this.submitHandler}>

                {form}

                <Button btnType="Success">Submit</Button>

                </form>

                <Button

                clicked={this.switchAuthModelHandler}

                btnType="Danger"

                >

                SWITCH TO {this.state.isSignup ?

                 'LOGIN' : 'SIGNUP'}</Button>

            </div>

        );

    }

}

const mapStateToProps = state =>{

    return {

        isAuthenticated: state.auth.token !== null,

        buildingBurger: state.burger.building,

        authRedirectPath: state.auth.authRedirectPath

    };

};

const mapDispatchToProps = dispatch => {

    return {

        onAuth: (email,password, isSignup) => dispatch(actions.auth(email,password,isSignup)),

        onSetAuthRedirectPath: () => dispatch(actions.setAuthRedirectPath('/'))

    };

}

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

**APP: IT WILL REDIRECT USER TO BURGER BUILDER AFTER SUBMITTING FORM.**

**IF USER MAKE ORDER WITHOUT LOGIN, THEN THEY WILL REDIRECT TO LOGIN FORM. AFTER LOGIN THEY WILL REDIRECT TO CHECKOUT PAGE.**

# PERSISTENT AUTH STATE WITH LOCALSTORAGE [logout user if not authenticated or login user id authenticated for certain time]

Auth.js[actions folder]

      let url = 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        if(!isSignup){

            url='https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        }

        axios.post(url,authData)

        .then(response => {

            console.log(response);

            const expirationData = new Date(new Date().getTime() + response.data.expiresIn\*1000)

            localStorage.setItem('token', response.data.idToken);

            localStorage.setItem('expirationDate',expirationData);

            dispatch(authSuccess(response.data.idToken, response.data.localId))

            dispatch(checkAuthTimeout(response.data.expiresIn));

        })

        .catch( err =>{

            console.log(err);

            dispatch(authFail(err))

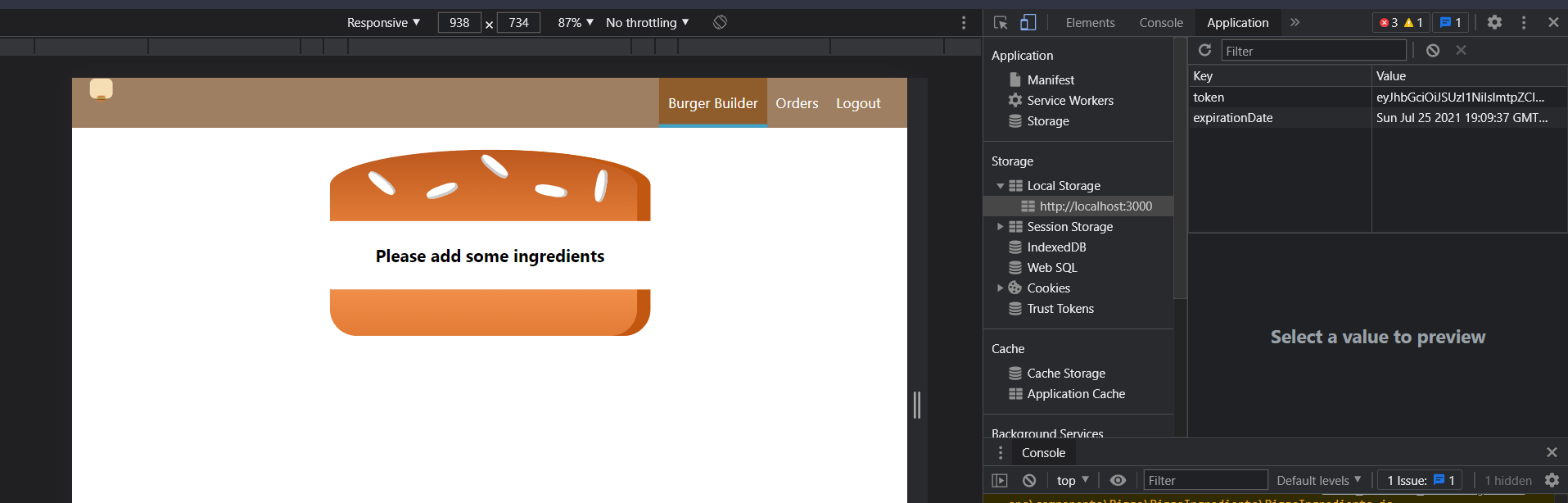
        })

    }

}

CONSOLE:

AFTER LOGIN



TO REMOVE AFTER LOGOUT:

Auth.js [actions folder]

export const logout = () => {

    localStorage.removeItem('token');

    localStorage.removeItem('expirationDate');

    //sync

    return{

        type: actionTypes.AUTH\_LOGOUT

    }

}

//APP.JS is root component which will loaded everytime. not matter which route we vist

//this makes great app to check authentication status

Auth.js [actions folder]

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

import axios from 'axios';

export const authStart = () => {

    return{

        type: actionTypes.AUTH\_START

    };

};

export const authSuccess = (token, userId) => {

    return{

        type: actionTypes.AUTH\_SUCCESS,

        idToken: token,

        userId: userId

    };

};

export const authFail = (error) => {

    return{

        type: actionTypes.AUTH\_FAIL,

        error: error

    };

};

//for logout

export const logout = () => {

    localStorage.removeItem('token');

    localStorage.removeItem('expirationDate');

    localStorage.removeItem('userId');

    //sync

    return{

        type: actionTypes.AUTH\_LOGOUT

    }

}

export const checkAuthTimeout = (expirationTIme) => {

    return dispatch =>{

        setTimeout(() => {

            //async

            dispatch(logout());

        // },expirationTIme);

    },expirationTIme\*1000); //for one hour

    };

};

//ASYNC CODE

export const auth = (email,password, isSignup) => {

    return dispatch => {

        //authenticate

        dispatch(authStart());

        const authData = {

            email: email,

            password: password,

            returnSecureToken: true

        };

        let url = 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        if(!isSignup){

            url='https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyDSmW9lJ7r\_nrKMc67\_iIhqgvgK17xJvug'

        }

        axios.post(url,authData)

        .then(response => {

            console.log(response);

            const expirationData = new Date(new Date().getTime() + response.data.expiresIn\*1000)

            localStorage.setItem('token', response.data.idToken);

            localStorage.setItem('expirationDate',expirationData);

            //storing user' data

            localStorage.setItem('userId', response.data.userId)

            dispatch(authSuccess(response.data.idToken, response.data.localId))

            dispatch(checkAuthTimeout(response.data.expiresIn));

        })

        .catch( err =>{

            console.log(err);

            dispatch(authFail(err))

        })

    }

}

export const setAuthRedirectPath = (path) => {

    return {

        type: actionTypes.SET\_AUTH\_REDIRECT\_PATH,

        //payload

        path: path

    }

}

export const authCheckState = () =>{

    return dispatch => {

        const token = localStorage.getItem('token');

        if(!token){

            dispatch(logout());

        } else {

            const expirationDate = new Date(localStorage.getItem('expirationDate'));

            if(expirationDate > new Date()){

                //we want to login

                const userId = localStorage.getItem('userId');

                dispatch(authSuccess(token,userId));

                dispatch(checkAuthTimeout(expirationDate.getSeconds() - new Date().getSeconds()));

            }else{

                dispatch(logout());

            }

        }

    };

};

EXPORT actions functions

Index.js[actions folder]

export {

    addIngredient,

    removeIngredient

} from './pizzaBuilder';

export {} from './order';

export {

    auth,

    logout,

    setAuthRedirectPath,

    authCheckState

} from './auth'

App.js

import React from 'react';

import Layout from '../hoc/Layout/Layout';

import PizzaBuilder  from './PizzaBuilder/PizzaBuilder';

import Checkout from './Checkout/Checkout';

import Orders from './Orders/Orders';

import { Route,Switch } from 'react-router-dom';

import Auth from './Auth/Auth';

import Logout from './Auth/logout/logout';

import {connect } from 'react-redux';

import \* as actions from '../store/actions/index'

class App extends React.Component{

  componentDidMount(){

    this.props.onTryAutoSignup();

  }

  render(){

    return(

      <div className="App">

        <Layout>

          <Switch>

            <Route path="/checkout" component={Checkout} />

            <Route path="/orders" component={Orders} />

            <Route path="/auth" component={Auth} />

            <Route path="/logout" component={Logout} />

            <Route path="/" exact component={PizzaBuilder} />

          </Switch>

        </Layout>

      </div>

    )

  }

}

//APP.JS is root component which will loaded everytime. not matter which route we vist

//this makes great app to check authentication

const mapDispatchToProps =  dispatch => {

  return{

    onTryAutoSignup: () => dispatch(actions.authCheckState())

  };

};

export default connect(null,mapDispatchToProps)(App);

OR ELSE  
import React,{useEffect} from 'react';

import Layout from '../hoc/Layout/Layout';

import PizzaBuilder  from './PizzaBuilder/PizzaBuilder';

import Checkout from './Checkout/Checkout';

import Orders from './Orders/Orders';

import { Route,Switch } from 'react-router-dom';

import Auth from './Auth/Auth';

import Logout from './Auth/logout/logout';

import {connect } from 'react-redux';

import \* as actions from '../store/actions/index'

const App = (props) => {

  useEffect(() => {

    props.onTryAutoSignup();

  })

  return(

    <div className="App">

      <Layout>

        <Switch>

          <Route path="/checkout" component={Checkout} />

          <Route path="/orders" component={Orders} />

          <Route path="/auth" component={Auth} />

          <Route path="/logout" component={Logout} />

          <Route path="/" exact component={PizzaBuilder} />

        </Switch>

      </Layout>

    </div>

  )

}

//APP.JS is root component which will loaded everytime. not matter which route we vist

//this makes great app to check authentication

const mapDispatchToProps =  dispatch => {

  return{

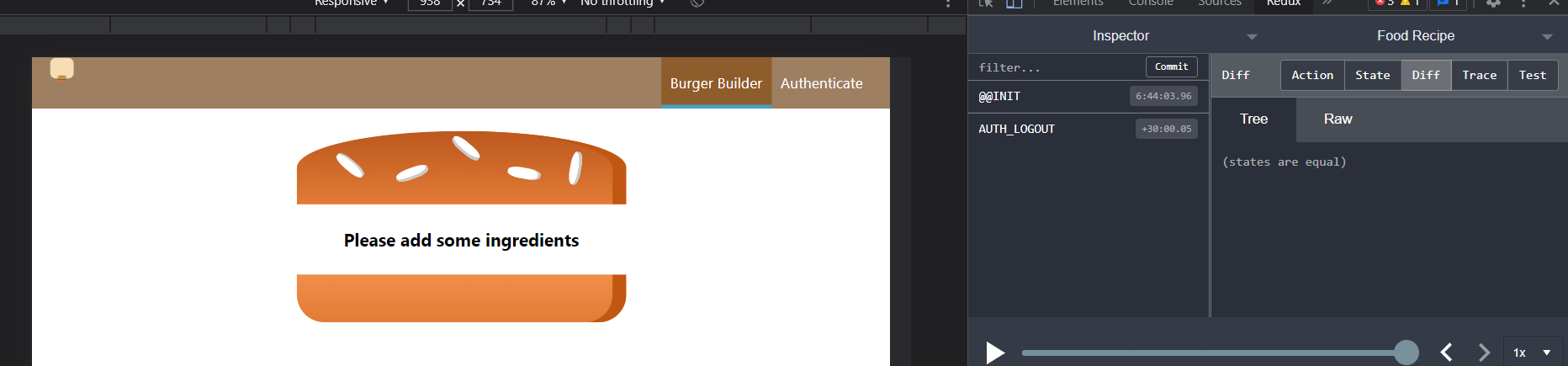
    onTryAutoSignup: () => dispatch(actions.authCheckState())

  };

};

export default connect(null,mapDispatchToProps)(App);

APP:



# FIXING CONNECT + ROUTING ERRORS []

WithRouter to fix routing error

import React,{useEffect} from 'react';

import Layout from '../hoc/Layout/Layout';

import PizzaBuilder  from './PizzaBuilder/PizzaBuilder';

import Checkout from './Checkout/Checkout';

import Orders from './Orders/Orders';

import { Route,Switch, withRouter } from 'react-router-dom';

import Auth from './Auth/Auth';

import Logout from './Auth/logout/logout';

import {connect } from 'react-redux';

import \* as actions from '../store/actions/index'

const App = (props) => {

  useEffect(() => {

    props.onTryAutoSignup();

  })

  return(

    <div className="App">

      <Layout>

        <Switch>

          <Route path="/checkout" component={Checkout} />

          <Route path="/orders" component={Orders} />

          <Route path="/auth" component={Auth} />

          <Route path="/logout" component={Logout} />

          <Route path="/" exact component={PizzaBuilder} />

        </Switch>

      </Layout>

    </div>

  )

}

//APP.JS is root component which will loaded everytime. not matter which route we vist

//this makes great app to check authentication

const mapDispatchToProps =  dispatch => {

  return{

    onTryAutoSignup: () => dispatch(actions.authCheckState())

  };

};

export default withRouter(connect(null,mapDispatchToProps)(App));

**EVEN AFTER RELOADING, USER DON’T GET LOG OUT BUT THEY WILL WHEN THEY CLICK LOG OUT BUTTON.**

export const authCheckState = () =>{

    return dispatch => {

        const token = localStorage.getItem('token');

        if(!token){

            dispatch(logout());

        } else {

            const expirationDate = new Date(localStorage.getItem('expirationDate'));

            if(expirationDate > new Date()){

                //we want to login

                const userId = localStorage.getItem('userId');

                dispatch(authSuccess(token,userId));

                dispatch(checkAuthTimeout((expirationDate.getTime() - new Date().getTime()) / 1000 ));

            }else{

                dispatch(logout());

            }

        }

    };

};

# ENSURING APP SECURITY

React will gives user security. RefreshTOken can be access through http json request.

# GAURDING ROUTES

## Adding user names in orders list so, only specific user can have access to their own orders.

## Only visit checkout and order page if we make order.

App.js

import React,{useEffect} from 'react';

import Layout from '../hoc/Layout/Layout';

import PizzaBuilder  from './PizzaBuilder/PizzaBuilder';

import Checkout from './Checkout/Checkout';

import Orders from './Orders/Orders';

import { Route,Switch, withRouter } from 'react-router-dom';

import Auth from './Auth/Auth';

import Logout from './Auth/logout/logout';

import {connect } from 'react-redux';

import \* as actions from '../store/actions/index'

const App = (props) => {

  useEffect(() => {

    props.onTryAutoSignup();

  })

  let routes =(

    //JSX

    <Switch>

    <Route path="/auth" component={Auth} />

    <Route path="/" exact component={PizzaBuilder} />

    </Switch>

  );

  if(props.isAuthenticated){

    routes=(

      <Switch>

       <Route path="/checkout" component={Checkout} />

       <Route path="/orders" component={Orders} />

       <Route path="/logout" component={Logout} />

        <Route path="/" exact component={PizzaBuilder} />

      </Switch>

    );

  }

  return(

    <div className="App">

      <Layout>

          {routes}

      </Layout>

    </div>

  )

}

//APP.JS is root component which will loaded everytime. not matter which route we vist

//this makes great app to check authentication

const mapStateToProps = state =>{

  return{

    isAuthenticated: state.auth.token !=null

  };

};

const mapDispatchToProps =  dispatch => {

  return{

    onTryAutoSignup: () => dispatch(actions.authCheckState())

  };

};

export default withRouter(connect(mapStateToProps,mapDispatchToProps)(App));

TO REDIRECT TO HOME PAGE

import { Route,Switch, withRouter,Redirect } from 'react-router-dom';

 let routes =(

    //JSX

    <Switch>

    <Route path="/auth" component={Auth} />

    <Route path="/" exact component={PizzaBuilder} />

    <Redirect to="/" />

    </Switch>

  );

if(props.isAuthenticated){

    routes=(

      <Switch>

       <Route path="/checkout" component={Checkout} />

       <Route path="/orders" component={Orders} />

       <Route path="/logout" component={Logout} />

       <Route path="/" exact component={PizzaBuilder} />

      <Redirect to="/" />

      </Switch>

    );

  }

# DISPLAYING USER SPECIFIC ORDERS