**24. Bonus Working with NgRx in our Project – Notes**

**16. First Summary & Clean Up**

**1. Add ngrx to app:**

 imports: [

     StoreModule.forRoot({ shoppingList: shoppingListReducer }),

  ],

**Tells ngrx which reducers we have in our application.**

**2. reducer is just a function.**

export function shoppingListReducer(

**Set State and Actions.**

  state: State = initialState,

  action: ShoppingListActions.ShoppingListActions) {

**3. Switch on action type.**

switch (action.type) {

    case ShoppingListActions.ADD\_INGREDIENT:

**4. In actions file, set unique identifiers for all actions.**

export const ADD\_INGREDIENT = 'ADD\_INGREDIENT';

**Each action is a class with a type property and a maybe a constructor where we can attach data to that action.**

export class AddIngredient implements Action {

    readonly type = ADD\_INGREDIENT;

    constructor(public payload: Ingredient) { }

}

**5. Can remove ShoppingListService completely from the application.**

**17. One Root State**

**1. Add auth/store/auth.reducer.ts**

**Define function**

export function authReducer(state = initialState, action) {

    return state;

}

**Set initial state**

const initialState : State = {

    user:null

}

**Create interface to describe State**

export interface State{

    user:User

}

**2. Add reducer to app.module.ts**

    StoreModule.forRoot({ shoppingList: shoppingListReducer, auth: authReducer })

**3. Add global store folder (app/store)with app.reducer.ts**

**Move AppState interface from shoppingListReducer to app.reducer.ts.**

**Create ActionReducerMap**

export const appReducer: ActionReducerMap<AppState> = {

    shoppingList: fromShoppingList.shoppingListReducer,

    auth: fromAuth.authReducer

};

**Replace ActionReducerMap in app.module.ts**

**4. Replace fromShoppingList in shopping-list.component.ts**

import \* as fromApp from '../store/app.reducer';

 private store: Store<fromApp.AppState>

**5. Replace fromShoppingList in ShoppingEditComponent.**

    private store: Store<fromApp.AppState>

**6. Replace fromShoppingList in ShoppingEditComponent.**

private store: Store<{ shoppingList: fromApp.AppState }>) { }

**18. Setting Up Auth Reducer & Actions**

**1. in auth.service.ts wherever we have user.next(), we want to use ngrx and dispatch an action.**

**2. Add auth.actions.ts to auth store**

**Add consts and classes.**

**Add constructor. Payload is a User object.**

import { Action } from "@ngrx/store";

export const LOGIN = 'LOGIN';

export const LOGOUT = 'LOGOUT';

export class Login implements Action {

    readonly type = LOGIN;

    constructor(public payload: {

        email: string,

        userid: string,

        token: string,

        expirationDate: Date

    }

    ) { }

}

export class Logout implements Action {

    readonly type = LOGOUT;

}

**3. export actions**

export type AuthActions = Login | Logout;

**4. auth.reducer.ts**

**Set type on action.**

**Switch on action.types.**

export function authReducer

    (state = initialState,

        action: AuthActions.AuthActions) {

    switch (action.type) {

        case AuthActions.LOGIN:

        case AuthActions.LOGOUT:

        default:

            return state;

    }

**5. Add Login and Logout functionality.**

  case AuthActions.LOGIN:

            const user = new User(

                action.payload.email,

                action.payload.userid,

                action.payload.token,

                action.payload.expirationDate

            );

            return {

                ...state,

                user: user

            }

        case AuthActions.LOGOUT:

            return {

                ...state,

                user: null

            }