**NGRX NOTES**

A typical NGRX setup consists of the following elements:

* Store: this is were all the data is stored, and you cannot manipulate the store directly. Using the Redux DevTools, you can inspect the data in the store in your browser.
* Action: defines all state changing actions that are available
* Reducer: where you define what the store should do in response to an action. It takes a state and an action as arguments, and returns a new state.
* Effect: a function that is triggered after an action is performed.
* Selector: a function that allows you to access the data in the store

The store is defined in the project Module file (using StoreModule.forRoot() in the main module or StoreModule.forFeature() in feature modules) and sets the name for the store and the reducers. For example:

@NgModule({

imports: [

StoreModule.forFeature(‘courses’, coursesReducer)

The reducer is created using the createReducer() function and takes an initial state, and separate functions for all the actions that will change the state. These functions only return the new modified state, for example:

export const authReducer = createReducer(

initialAuthState,

on(AuthActions.login, (state, action) => { user: action.user }

);

Many helper functions exist for reducers, such as NGRX’s Entity Adapter or several form methods (like formStateReducer or validate functions).

Reducers react on actions, and an action is a simple definition created using NGRX’s createAction() function. It takes a type (i.e. a name for the action) and optionally a props object (i.e. a model specifying what type of data is passed through the action). For example:

export const login = createAction(

‘[Login Component] User login’,

props<{ user: User }>()

);

The effect is a function that is triggered on each action. In the effects, actual operations are performed with the data passed through an action, such as fetching data from a service, storing data to the localStorage or navigating to another page.