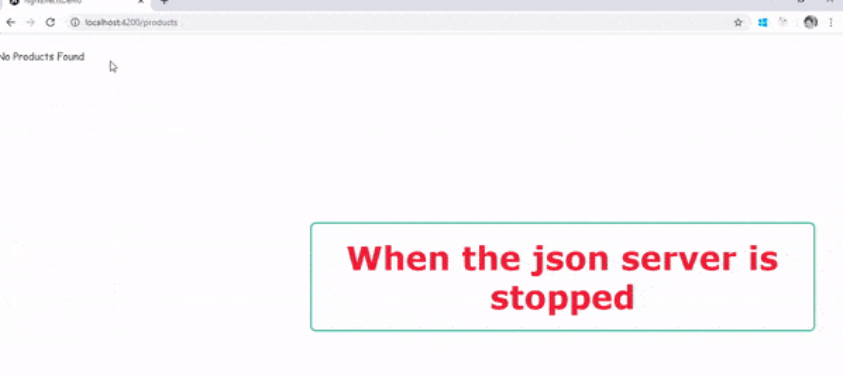
**Problem Statement:-**Create a component **product-list**that uses **ngRx effects** to make API call for loading the list of products is success case and displays error message is if data not found as shown below:





To achieve the above requirement, Follow the below steps:

**1.**Create an angular app and install **@ngrx/effects** in the angular application:

1. npm install @ngrx/effects --save

**2.**Define a model class for the Products data in the file **product-list/products.ts:**

1. export class Product {
2. id: number=0;
3. productName: string="";
4. productPrice: number=0;
5. productImg: string="";
6. productBrand: string="";
7. }

**3.**Define the **actions**for handling both success and error cases of API calls in the file **Store/actions/products.actions.ts:**

1. import { createAction, props } from '@ngrx/store';
2. import { Product } from '../../product-list/products';
3. export const getAllProducts = createAction('[ProductList Component] GET\_ALL\_PRODUCTS')
4. export const getAllProductsAPISuccess = createAction('[ProductList Component] GET\_ALL\_PRODUCTS SUCCESS', props<{ allProducts: Product[] }>())
5. export const getAllProductsAPIError = createAction('[ProductList Component] GET\_ALL\_PRODUCTS ERROR', props<{ errorMessage: String }>())

**getAllProducts** is the action to be dispatched on **load**of**product-list** component, whereas **getAllProductsAPISuccess**and **getAllProductsAPIError**are the actions dispatched by the Effect once it completes the API/service call with success or error response

**4.**Define the **Initial state**for the application in the file **Store/states/products.store.ts:**

1. export const initialProductState = {
2. allProducts: [],
3. errorMessage: ""
4. }

**allProducts**is an array that will store an array of product objects fetched from service call whereas **errorMessage**will store the error response in case the service call does not return the required data.

**5.**Define the **reducer**for handling both success and error cases of API calls in the file **Store/reducers/products.reducer.ts:**

1. import { createReducer, on } from '@ngrx/store';
2. import { getAllProducts, getAllProductsAPISuccess, getAllProductsAPIError } from '../actions/products.actions';
3. import { initialProductState } from '../states/products.store';
4. const \_productReducer = createReducer(initialProductState,
5. on(getAllProducts, (state) => state),
6. on(getAllProductsAPISuccess, (state:any, { allProducts }) => { return { ...state, allProducts: allProducts, errorMessage: "" } }),
7. on(getAllProductsAPIError, (state:any, { errorMessage }) => { return { ...state, errorMessage: errorMessage, allProducts: [] } })
8. );
9. export function productReducer(state:any, action:any) {
10. return \_productReducer(state, action);
11. }

**productReducer**is the reducer function that handles actions dispatched by the components as well as the effects after making the API calls

**6.**Define the selectors in the file **Store/selector/products.selector.ts** to ensure the component gets only the required data:

1. import { createSelector } from '@ngrx/store';
2. import { Product } from '../../product-list/products';
3. export interface FeatureState {
4. allProducts: Product[];
5. errorMessage: String;
6. }
7. export interface AppState {
8. products: FeatureState;
9. }
10. export const selectFeature = (state: AppState) => { return state.products };
11. export const selectError = (state: AppState) => { return state.products };
12. export const selectFeatureProduct = createSelector(
13. selectFeature,
14. (state: FeatureState) => state.allProducts
15. );
16. export const selectFeatureError = createSelector(
17. selectError,
18. (state: FeatureState) => state.errorMessage
19. );

**FeatureState**defines the structure of the data stored in the **store**object

**selectFeatureProduct**will return the products array from the global array whereas **selectFeatureError**will return the errorMessage set by the reducer

**7.**Create the **Effects**that invokes the appropriate service method to fetch all the products from server:

1. import { Injectable } from '@angular/core';
2. import { Actions, createEffect, ofType } from '@ngrx/effects';
3. import { ProductService } from '../../product-list/product.service';
4. import { map, mergeMap, catchError } from 'rxjs/operators';
5. @Injectable()
6. export class ProductEffects {
7. constructor(private actions$: Actions, private productService: ProductService) { }
8. loadProducts$ = createEffect(() => this.actions$.pipe(
9. ofType('[ProductList Component] GET\_ALL\_PRODUCTS'),
10. mergeMap(() => this.productService.getAllProducts()
11. .pipe(
12. map(products => ({ type: '[ProductList Component] GET\_ALL\_PRODUCTS SUCCESS', allProducts: products })),
13. catchError(() => of({ type: '[ProductList Component] GET\_ALL\_PRODUCTS ERROR', errorMessage: 'No Products Found' }))
14. ))));
15. }

**createEffect**is the method used to define an Effect, as discussed previously.

**8.**Create a Service **product** with the following code to fetch all the products from the server:

1. import { Injectable } from '@angular/core';
2. import { HttpClient } from '@angular/common/http';
3. import { Observable } from 'rxjs';
4. import { Product } from './products';
5. @Injectable({
6. providedIn: 'root'
7. })
8. export class ProductService {
9. constructor(private http: HttpClient) { }
10. getAllProducts(): Observable<Product[]> {
11. return this.http.get<Product[]>("http://localhost:4000/products");
12. }
13. }

**getAllProducts()**is a servicemethod which makes an API call to fetch all the products data from the service

**9.**Create a **product-list**component and add the following code to dispatch an action to fetch all the products on load of the component:

1. import { Component, OnInit } from '@angular/core';
2. import { Product } from './products';
3. import { Store, select } from '@ngrx/store';
4. import { Observable } from 'rxjs';
5. import \* as fromRoot from '../Store/selector/products.selector';
6. @Component({
7. selector: 'app-product-list',
8. templateUrl: './product-list.component.html',
9. styleUrls: ['./product-list.component.css']
10. })
11. export class ProductListComponent implements OnInit {
12. constructor(private store: Store<fromRoot.AppState>) { }
13. products$: Observable<Product[]> = this.store.pipe(select(fromRoot.selectFeatureProduct))
14. errorMessage$: Observable<String> = this.store.pipe(select(fromRoot.selectFeatureError))
16. ngOnInit(): void {
17. this.store.dispatch({ type: '[ProductList Component] GET\_ALL\_PRODUCTS' });
18. }
19. }

**products$**is the property that stores the products array fetched from the global store. **errorMessage$** is the property that stores the error message fetched from the global store. An action of type **'[ProductList Component] GET\_ALL\_PRODUCTS'**is dispatched on load of component using **ngOnInit()** method.

**10.**Add the following code to **product-list.component.html** file:

1. <div class="row mt-4 text-center" style="font-family: cursive;">
2. <div class="col-md-3" \*ngFor="let product of products$ | async; let i = index">
3. <div class="card shadow">
4. <img src="{{product.productImg}}" alt="{{product.productName}}">
5. <div class="card-body">
6. <h5>{{product.productName}}</h5>
7. <h6>Brand: {{product.productBrand}}</h6>
8. <h6 class="text text-success">Price: {{product.productPrice | currency:'INR'}}</h6>
9. </div>
10. </div>
11. </div>
12. <span \*ngIf="errorMessage$">{{errorMessage$ | async}}</span>
13. </div>

**asyc**pipe is used for both **products$**and **errorMessage$** as it unwraps a value from an asynchronous primitive

**11.**Configure the **effects** in **app.module.ts**file:

1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3. import { AppRoutingModule } from './app-routing.module';
4. import { AppComponent } from './app.component';
5. import { ProductListComponent } from './product-list/product-list.component';
6. import { HttpClientModule } from '@angular/common/http';
7. import { StoreModule, ActionReducer, MetaReducer } from '@ngrx/store';
8. import { productReducer } from './Store/reducers/products.reducer';
9. import { EffectsModule } from '@ngrx/effects';
10. import { ProductEffects } from './Store/effects/products.effects';
11. export function debug(reducer: ActionReducer<any>): ActionReducer<any> {
12. return function (state, action) {
13. console.log('previous state', state);
14. console.log('action', action);
15. let nextState = reducer(state, action);
16. console.log('current state', nextState);
17. return nextState;
18. };
19. }
20. export const metaReducers: MetaReducer<any>[] = [debug];
21. @NgModule({
22. declarations: [
23. AppComponent,
24. ProductListComponent
25. ],
26. imports: [
27. BrowserModule,
28. AppRoutingModule,
29. HttpClientModule,
30. StoreModule.forRoot({ products: productReducer }, { metaReducers }),
31. EffectsModule.forRoot([ProductEffects])
32. ],
33. providers: [],
34. bootstrap: [AppComponent]
35. })
36. export class AppModule { }

**EffectsModule.forRoot()** takes an array of effects to be used in the angular application to make API calls

**12.** Define the **routes** in**app-routing.module.ts**and load the **product-list** component by adding **router-outlet** in**app.component.html:**

1. const routes: Routes = [
2. { path: '', redirectTo: 'products', pathMatch: 'full' },
3. { path: 'products', component: ProductListComponent }
4. ];

**13.** To access the server and fetch the data from the server, follow the below steps: **[This is just for demonstration purpose, you can use any other server such as express web service as well, in that case below steps are not required to be followed]**

* Install **json-server** globally and locally:

1. npm install json-server -g
2. npm install json-server --save

* Create a **db.json** file in project root folder and and add the following data:

1. {
2. "products": [
3. {
4. "id": 1001,
5. "productName":"iPhone 7s",
6. "productPrice":39999,
7. "productImg":"iphone.jpg",
8. "productBrand":"Apple"
9. },
10. {
11. "id": 1002,
12. "productName":"Nokia 8",
13. "productPrice":24499,
14. "productImg":"nokiasmartphone.png",
15. "productBrand":"Nokia"
16. },
17. {
18. "id": 1003,
19. "productName":"Redmi Note 8 Pro",
20. "productPrice":39999,
21. "productImg":"redminote8.PNG",
22. "productBrand":"Xiaomi"
23. },
24. {
25. "id": 1004,
26. "productName":"Samsung Galaxy s10",
27. "productPrice":39999,
28. "productImg":"samsunggalaxy.jpg",
29. "productBrand":"Samsung"
30. }
31. ]
32. }

* Run the json-server using the below command in project folder: