## **1. 🧭 Overview**

This Angular application serves as a starter template for apps requiring user authentication, route protection, and state-managed data retrieval. It uses:

* **Angular 19** (frontend framework)
* **NgRx** (state management)
* **Bootstrap 5** (UI design)
* **FakeStoreAPI** (mock product data)

## **2. 🏗️ Architecture Diagram**

**Core Modules:**

sql

CopyEdit

[ Auth ]

|--> LoginComponent

|--> SignupComponent

|--> AuthService

|--> NgRx: auth.actions, auth.reducer, auth.effects

[ Product ]

|--> ProductListComponent

|--> ProductService

|--> NgRx: product.actions, product.reducer, product.effects

[ Core ]

|--> AuthGuard

[ App Module ]

|--> Routing, StoreModule, EffectsModule

## **3. 📦 Feature Modules**

### **🔐 Auth Module**

| **Feature** | **Description** |
| --- | --- |
| Login | User authentication via form |
| Signup | Register a new user |
| Auth Effects | Side effects for login/signup |
| State | Tracks authentication status and user info |

### **🛒 Product Module**

| **Feature** | **Description** |
| --- | --- |
| Product List | Fetch and display list of products |
| Product Effects | API interaction for product data |
| State | Product list and loading status |

## **4. 🔄 State Management Design (NgRx)**

### **Auth Store**

* **Actions**: login, loginSuccess, loginFailure, signup, logout
* **Selectors**: selectIsAuthenticated, selectUser

**Reducer State**:  
  
{

user: User | null,

token: string | null,

isAuthenticated: boolean,

error: string | null

}

### **Product Store**

* **Actions**: loadProducts, loadProductsSuccess, loadProductsFailure
* **Selectors**: selectProductList

**Reducer State**:  
  
  
{

products: Product[],

loading: boolean,

error: string | null

}

## **5. 🔐 Security & Access Control**

* **Route Guard** (AuthGuard) checks token in localStorage.
* Routes are protected by canActivate.
* User is redirected to /login if unauthenticated.

## **6. 🌐 Routing Table**

| **Route** | **Component** | **Guarded** | **Description** |
| --- | --- | --- | --- |
| / | Redirect to /login | No | Default route |
| /login | LoginComponent | No | Login page |
| /signup | SignupComponent | No | Signup page |
| /home | ProductListComponent | ✅ Yes | Protected product list |

## **7. 🎨 UI Guidelines (Bootstrap)**

* **Grid Layout**: Bootstrap framework.
* **Forms**: form-control, form-group, and validation classes.
* **Feedback**: Alerts used for error/success messages.

## **8. 🔌 API Integration**

* Fake Store API: https://fakestoreapi.com/products
* Integrated via HttpClient in ProductService.
* Handled through ProductEffects.

## **9. 💾 Local Storage Use**

Stored on successful login:

* auth\_token
* user

Used by:

* AuthService
* AuthGuard

## **10. 🚀 Bootstrapping Process**

1. App launches -> redirects to /login

2. LoginComponent renders form

3. User logs in -> NgRx dispatches login

4. AuthEffects stores token -> navigates to /home

5. HomeComponent loads -> dispatches loadProducts

6. ProductEffects calls API -> populates product store

## **11. 🧪 Testing Strategy**

| **Component** | **Testing Type** |
| --- | --- |
| Forms | Unit test form validations |
| Effects | Jasmine/Karma for side effects |
| Reducers | Pure function tests |
| Guard | AuthGuard logic via service mocking |
| UI | Manual E2E test for flow verification |

## **12. 📈 Scalability Notes**

* Each module is isolated, allowing feature-based scaling.
* NgRx state scales horizontally with new domains.
* Reusable guards and services.