**Product Application**

### Document Properties

| Item | Details |
| --- | --- |
| Document | Architecture Design Document |
| Application | Product Application |
| Author | SURAJ MONGAD |
| Version | V 1.0 |
| Creation Date | 14/9/2020 |
| Last Updated | 15/9/2020 |
| Status | Completed |

Contents

[**1.** **Introduction**](#_Toc45989189)

[**2.** **Architecture Diagram**](#_Toc45989190)

[**3.** **Use-Case View**](#_Toc45989191)

[**4.** **Flow Chart**](#_Toc45989192)

[**5.** **Components**](#_Toc45989193)

[**6.** **Services**](#_Toc45989194)

**7**. **NGRX**

**8. Responsive web design (RWD)**

[**9.** **Validations**](#_Toc45989195)

[**10.** **Tools and Libraries**](#_Toc45989196)

**11. Application View**

[**12.** **Unit Testing**](#_Toc45989197)

[**13.** **Accessibility**](#_Toc45989199)

[**14.** **Setup**](#_Toc45989201)

# Introduction

This document provides a comprehensive architectural overview of the application, using number of different architectural views to depict different aspects of the application.

# Architecture Diagram

index.html

<app-root></app-root>

Root Router

Services

Components

NgRx

Bootstrapping Application

Root Module

# Use-Case View

The application use cases are -

* **Public Listing** – Public listing means, display all the data of products and their associated data.
* **Login** – Users can login using their own account. Users can enter a username and password.
* **Signup –** If the user not having any account that time user can register there account and login to the account.
* **Add product to the list** – Users can create new products using the create button and he can add the product details in product input fields.
* **Edit product** - Any product can be editable by clicking on the edit button.
* **Delete product** – Each product will have a delete button. The product can be deleted by clicking on it.
* **Change theme**- App having a theme. Using that theme button user can change the theme of the application.
* **Logout** – Users can logout the application using logout button.

These use cases are initiated by the user.

# Flow Chart

Default Product page without login

Default Product List

Product page with login

Add and Edit Page

Product List Page with Add and Edit Button

Add and Edit Product Page.

# Components

* AppComponent – This component is used for Application Bootstrapping.
* LandingComponent – Landing page of applications that show all the list and their respective product with edit and delete feature for adding new product.
* ListProductComponent – This component contains form display product list and add and edit button.
* AddProductComponent – This component contain form for add product.
* EditProductComponent – This component contain form for edit product.

# Services

AuthService – This service generally having logic about product in the application and this contain code for getting all product list, login and signup with their edit product, add new product and delete product logic.

# NGRX

NGRX is a group of libraries “inspired” by the redux pattern which in turn is “inspired” by the Flux pattern. Being a little more concise, this means that redux pattern is a simplified version of the Flux pattern and NGRX is an angular/rxjs version of the redux pattern.

* **Store:** A store is an object (an instance of the ngrx Store class) that brings the things we mentioned before (Actions, Reducers, Selectors) together. For example, when an action is dispatched (using the store object dispatch function), the store is the one finding and executing the appropriate reducer.
* **Actions:** In the store object, you have a function to dispatch (trigger) actions. Actions are classes that implemenets the NGRX Action interface. These Action classes have two properties (let's take as an example an action class called GetUserName): type: it’s a read only string describing what the action stand for. For example: ‘[User] Get Username’. Payload: the type of this property depends on what type of data this action needs to send to the reducer. In the case of the previous example, is going to be a string containing the username. Not all actions need a payload.
* **Reducers**: Reducers are pure functions accepting two arguments, the previous state and an Action. When an Action is dispatched ngrx goes through all the reducers passing as arguments the previous state and the Action, in the order that the reducers where created, until it finds a case for that action.
* **Effects:** Effects, on the ngrx libraries ecosystem, allow us to deal with side-effects caused from dispatching an action outside angular component or the ngrx store.

The Effects listen if any action is dispatched, then, similar to what reducers do, it checks if the action is one of the actions type it has a case for. Then is going to perform a side-effect, usually getting or sending data to an API.

Finally, is going to emit another action, usually, an action referring to the result-state of the side effect (success, error, etc.), then a reducer is going to enter in the scene as we already mention in the ngrx flow.

# Responsive web design (RWD):

Responsive web design (RWD) is an approach to web design that makes web pages render well on a variety of devices and windows or screen sizes. We have used CSS grid structure for responsive web design

1. **Validations:**

1) Application covered the required validation for login page and sign-up.

2) Application covered the required validation for add page and edit.

# Tools and Libraries

- Angular version: 10.1.0

- Node version: 12.18.3

- TypesScript: 4.0.2

- Rxjs: 6.6.2

-webpack: 4.44.1

-jest: 26.4.2

-jest-preset-angular: 8.3.1

-protractor: 7.0.0

# Unit Testing

Added Unit Test cases for below file.

It consists of two services -

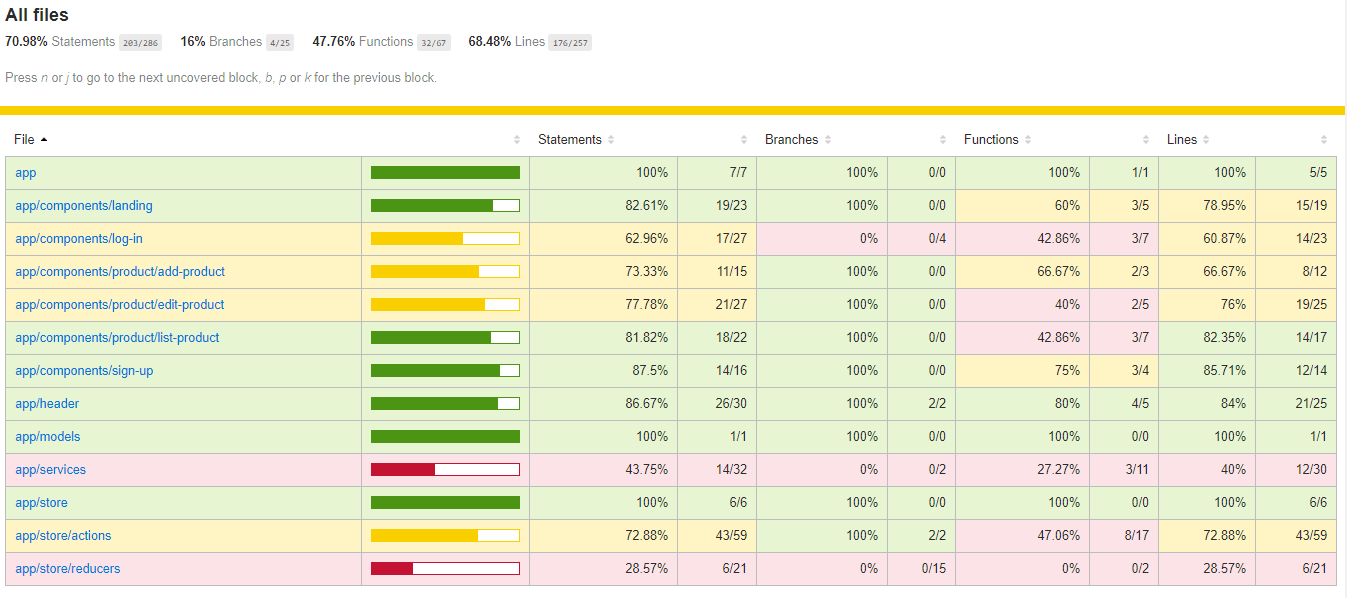
* + Product.action.spec.ts – It contain test cases for the all the logic like add product, update product, delete product and get product.
    - No of test cases: 7
  + Auth.actions.spect.ts- It contains test cases of login and signup.
    - No of test cases: 2

Component related test cases

* + - No of test cases: 11

**Total test cases: 20**

**Screenshot for unit test cases –**

****

# Accessibility

For Accessibility, application having appropriate HTML tags that are preferred for accessibility

User can run application via keyboard and tab.

# Setup

1. Download and install node.js (version: 12.18.3)
2. Install npm and install @angular (version: 10.1.0)
3. Git clone <https://github.com/smongad/tavisca_assignment.git> origin master
4. Navigate to cloned folder & run npm install
5. Run ng serve

For testing use: jest --collect-coverage