**NAGP Commerce**

**Angular Assignment**

Contents

[**Problem Statement** 3](#_Toc71017614)

[**Solution** 3](#_Toc71017615)

[**Important Point** 3](#_Toc71017616)

[**Assumptions** 4](#_Toc71017617)

[**Bonus Points Covered** 4](#_Toc71017618)

[**Project Links** 4](#_Toc71017619)

[**Data Storage Approach** 4](#_Toc71017620)

[**Authentication and Authorization Approach** 4](#_Toc71017621)

[**Unit Tests Files** 4](#_Toc71017622)

# **Problem Statement**

Develop an **“e-commerce”** web app using Angular (latest), HTML5, CSS3. Application should display list of the products. User should be able to search product, view details/description of product, add product to cart and checkout.

**Following are the key functional aspects:**

1. **Login Screen**: Validation. Create dummy user. Only authenticated user can go into the application.
2. **Search Implementation:** Create few dummy products and list products as per search criteria.
3. **Grid Product**: Everyone can see a grid page. On Click of product image/title user should be able to view **product description page.**

**Hint:** (Use reusable components here)

1. **Product Detailed Page:** The product description page **(PDP)** will have the details of the product such as **product** **name**, **price**, **description**, **tags**, **category**, **image**, **quantity** etc. All these attributes can be hardcoded for simplicity. **(no need to go into category details).**

The PDP screen will have an **add to cart button.** Upon clicking on add to cart, the user will be taken into a new screen which will have all the products being added to cart.

1. **Cart Screen**: Only **logged in** user can perform this action. Redirect to login screen if user is not logged in. The cart screen can **update the quantity of the product**, **delete a product**. The cart screen will show the **total of all the items** in the cart.

On clicking on checkout button, user will be redirected to the checkout screen.

1. **Checkout Screen:** Only logged in user can see this screen. Enter delivery details, name, shipping address, phone number, email. Add necessary validation on this form. Upon clicking on submit, show a message to the user “Order placed successfully.”
2. Write test cases for at least 1 component (not just html components but verifying the functionality as well) and 1 service.

**Good to have/ Bonus points:**

1. **Category Tree**: Try to classify the products under categories (build your data or mock json accordingly). Based on this hierarchy display the category tree for products.
2. **Translation**: For Example: System should support two languages at the moment for all the labels displayed on UI. In the header part we can have a dropdown to change the system language**.**

# **Solution**

## **Important Points**

1. The Netlify servers are very slow. So, its expected to have some latency on initial application load time.
2. A bit detailed doc is provided on the GitHub repo readme file where we can see gifs for features per screen.

## **Assumptions**

1. As cart and check screen should only be accessed by logged in user. I am assuming below flow in case of **Add to Cart** from Product details screen.

* If user is not logged in, on click of **Add to Cart** user will be navigated to login screen and after successful login, user will be navigated to **Cart Screen** and product will be visible in the cart.
* If user is already logged in, on click of **Add to Cart** product will be added to cart and user will be navigated to **Cart Screen**.

1. We have not created any user registration screen. So, application is for single user only. Although we can extend it easily.

## **Bonus Points Covered**

1. Category Tree
2. Multilingual Support
3. My Orders Screen

## **Project Links**

1. **GitHub:** <https://github.com/d33pakjangra/nagp-commerce>
2. **Deployed Application:** <https://608fec77575a240008a371d7--nervous-kare-db2ca0.netlify.app/>

**Username:** admin

**Password:** admin

## **Data Storage Approach**

We have used json for products and user data. And we have used **IndexedDb** throughout the application for replicating backend behaviour. Products and user are seed in indexed db on application load.

## **Authentication and Authorization Approach**

**Authentication:** We have validated user on authentication from users stored in indexed db and upon successful authentication, a key (isLoggedIn) is added in the local storage.

**Authorization:** We have created auth guard for authorizations of selected screens. The auth guard checks for the key in the local storage for authorization.

## **Unit Tests Files**

1. Login.component.ts
2. Auth.service.ts