UDACITY

PROJECT SPECIFICATION

MyStore

Best Practices

CRITERIA	MEETS SPECIFICATIONS
Scaffold and configure a single-page application with Angular	The project requires only `npm install` and `ng serve` to install and launch the application
Document the project in the README	The Project README includes a description of the project, as well as instructions for installation and launch
Organize and write clean code	Components, models, services, and other assets are easy to find and organized intuitively. Code is free from syntax errors and follows the Udacity Frontend Nanodegree Style Guide

CRITERIA	MEETS SPECIFICATIONS
Design for user experience	The experience and flow of the application resemble that of a typical e-commerce application, including: • The shopping cart page shows a total cost for all products in the cart • Input forms are validated (e.g., a minimum length for the customer's name on the shopping cart checkout page) • Feedback is given to the user when the cart is modified (i.e., how is the user alerted when a product has been added to the cart?) • The details page for a product shows a photo of the product, the name, the price, and the description. • Products can be removed from the cart • An order confirmation page (e.g. a "success" page) is shown to the user after successful checkout
Use CSS to style the application	All components are styled, and CSS syntax is clean and follows the Udacity Frontend Nanodegree Style Guide. The provided CSS may be used but is not required.

Components

CRITERIA	MEETS SPECIFICATIONS

CRITERIA	MEETS SPECIFICATIONS
Fetch and use data from an external API	The list of products is retrieved from an external API or by using the `HttpClient` service to read the provided `data.json` file. The product list template shows relevant product information to the user, including a photo of the product, the name, and the price. The `*ngFor` directive is used to loop over any lists.
Create a logical hierarchy of components	Application code clearly shows component hierarchy where child components are dependent on parent components, and sibling components are not dependent on each other.
Collect input from the user using controlled form elements and Angular events	`ngModel` is used on the element to bind a form control to a data property. `ngModelChange` is used to listen to any `ngModel` changes (i.e., rather than `change`).
Use Angular event bindings	The application listens for and responds to user actions (e.g., `click`) using Angular event bindings.

CRITERIA	MEETS SPECIFICATIONS
Create and use custom TypeScript models	The application includes a TypeScript model for a product. That is, component code incorporates instances of a `Product` type. Individual properties of the model are appropriately typed (e.g., `url: string`).

Data Flow

CRITERIA	MEETS SPECIFICATIONS
Use decorators to pass data between parent and child components	When information needs to be shared between parent and child components, the application uses the `@Input` decorator. Conversely, when sending data from a child component to its parent component, the application uses the `@Output` decorator and `EventEmitter` class.
Use a service to pass data between sibling components	Cart data is shared between the product list component and the shopping cart component. As such, a service is used to facilitate passing data between said components.

Routing

CRITERIA	MEETS SPECIFICATIONS
Use Angular routing in templates	The application uses the `<router-outlet>` placeholder and the `routerLink` attribute in HTML templates. As a single-page application, the page does not reload during navigation.
Set up and configure the app routing module	An `AppRoutingModule` is included in the application, with relevant components imported and `path` and `component` values appropriately set.

Suggestions to Make Your Project Stand Out!

1. Enable a signup/login flow using Auth0.
2. Incorporate the back end built in the second course of this Nanodegree program to persist data.
3. Apply your own styling to the application.
4. Show an item amount “badge” next to the link for your shopping cart. Additionally, show the properly-calculated cart total in the cart. When no items are in the cart, indicate that the cart is empty.