Capgemini

ANGULAR MASTER CLASS PARTICIPANT GUIDE

E-Commerce Application

Contents

Introduction

This workshop introduces you to the essentials of Angular by walking you through a simple e-commerce site with a catalog, shopping cart, and check-out form using Angular Cli 10. To help you get started right away, in this workshop we use a simple ready-made environment that you can extract and start the application development without installing any softwares and node packages.

By the end of this class you will be able to do the following:

- Create Angular components.
- Use built-in Angular directives to show and hide elements and display lists of items.
- Use one-way data binding for read-only data.
- Add editable fields to update a model with two-way data binding.
- Bind component methods to user events.
- Enable users to select an item from a master list and edit it in the details view.
- Format data with pipes.
- Create a shared services.
- Use routing to navigate among different views and their components.

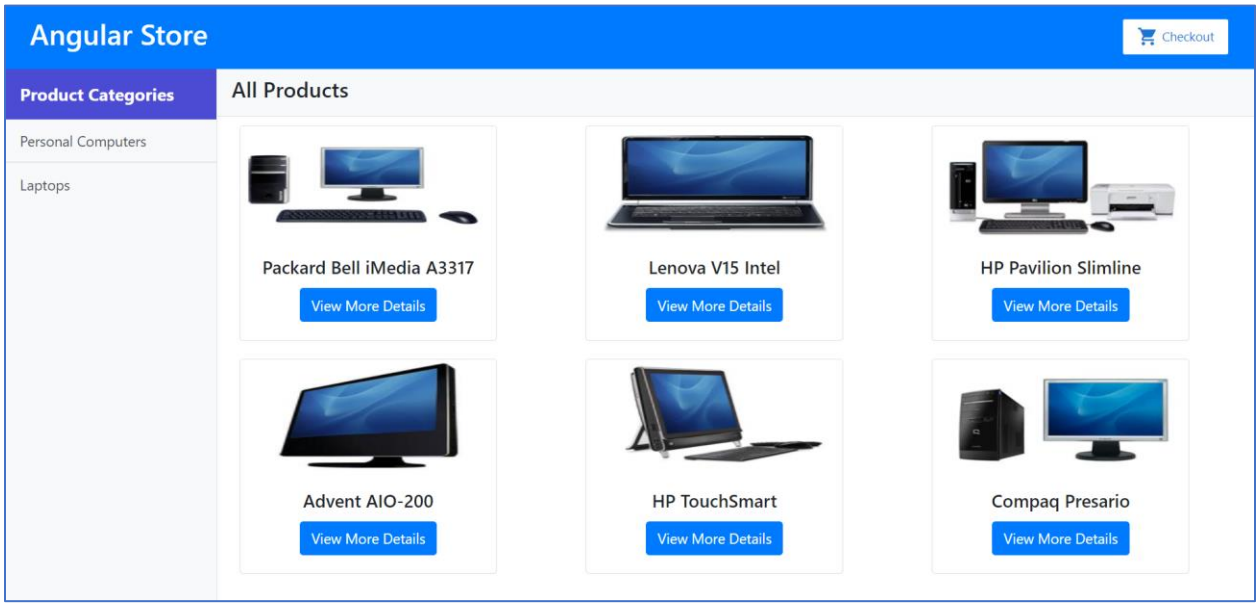
E-Commerce Application Walkthrough

This is a simple online store application with a product catalog, a shopping cart, and a checkout functionality.

Home Page:

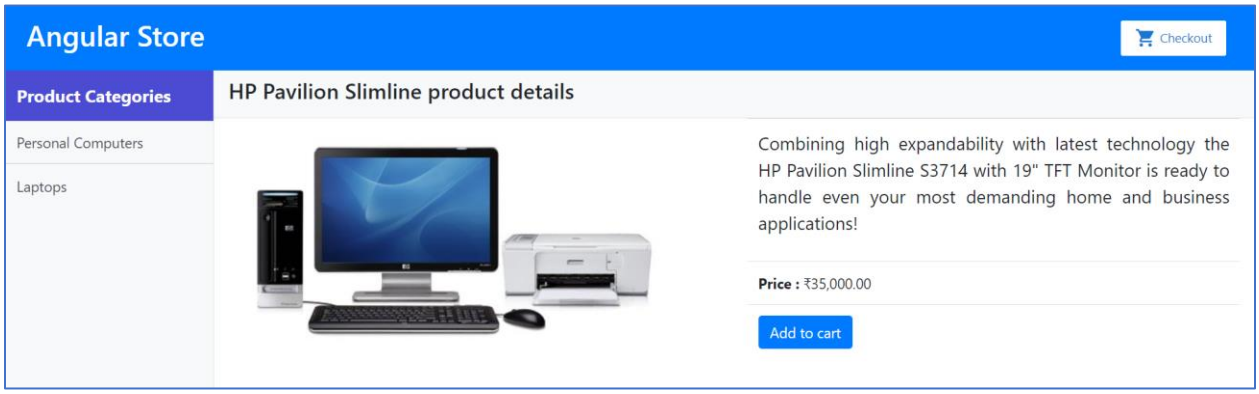
Home Screen contains the following

- Top Navigation Bar with Site Title and Checkout Button
- Side Bar in the Left Side will populates all the Product Categories
- Right side area acts as a place holder to display the Main Content of the Page. (By Default it displays all the Products Thumbnail images with View More Details Button)



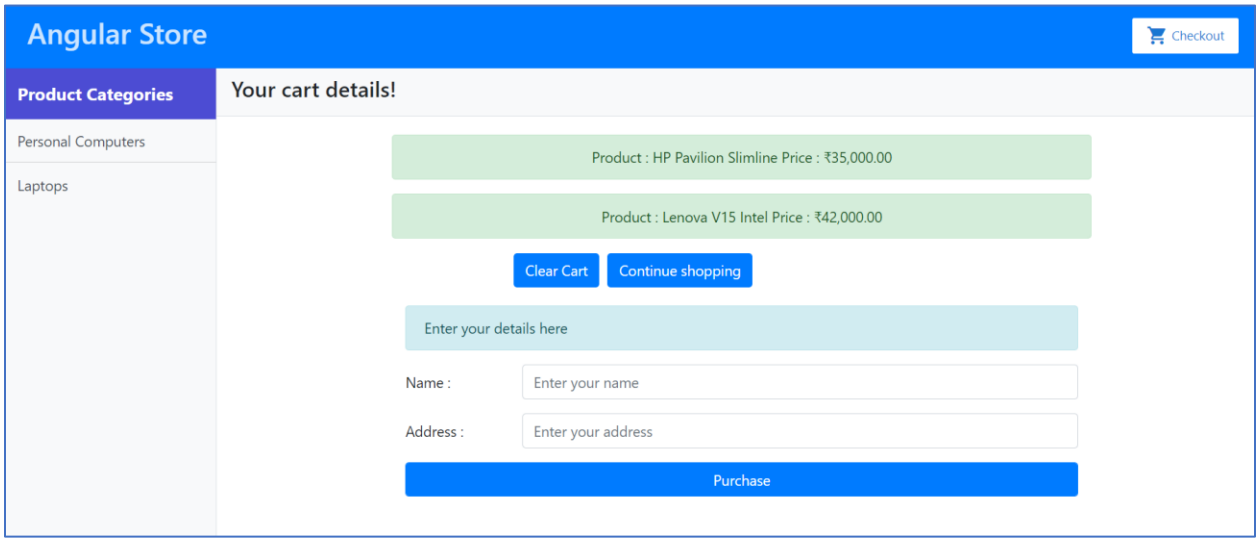
Product Details

Product Details Page displays the respective Product Information (ProductTitle, Image, Description & Price) when View More Details button in the product Gallery is clicked.



Checkout Page

Check out Page display the Items added in the cart and it also provides a form where customer can fill their name and address and purchase the items added in the cart.

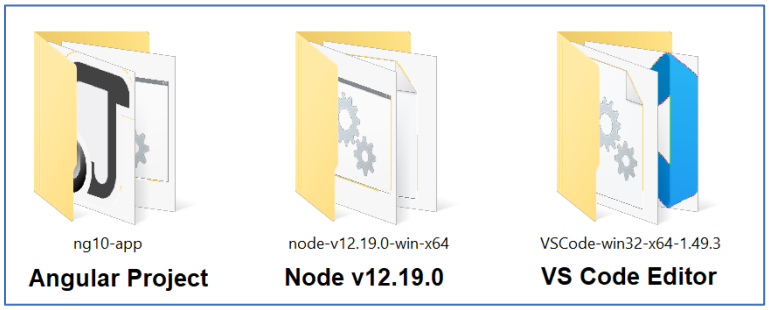


Environment Setup

Estimated Time : 15 mins

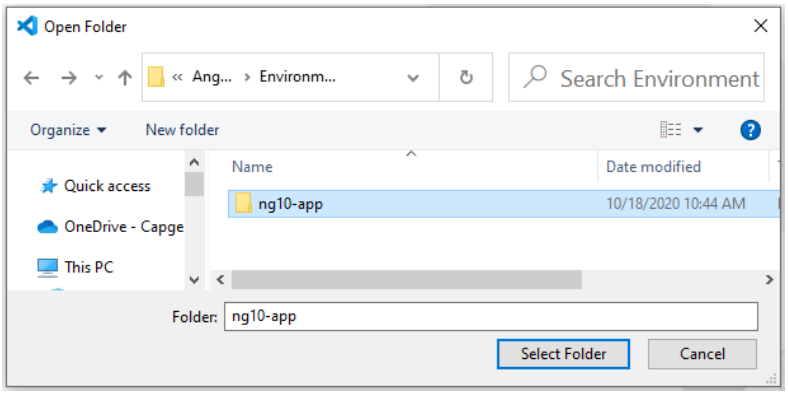
1. Download Environment.zip file from https://fts.capgemini.com/private/15742098350144/Environment.zip link valid until 2021-02-28 4:00 UTC (SSO protected)

2. Extract the Environment.zip file which has 3 folders
   a. **ng10-app** : Angular 10 Project
   b. **node-v12.19.0-win-x64** : Node Environment
   c. **VSCode-win32-x64-1.49.3** : Visual Studio Code Editor

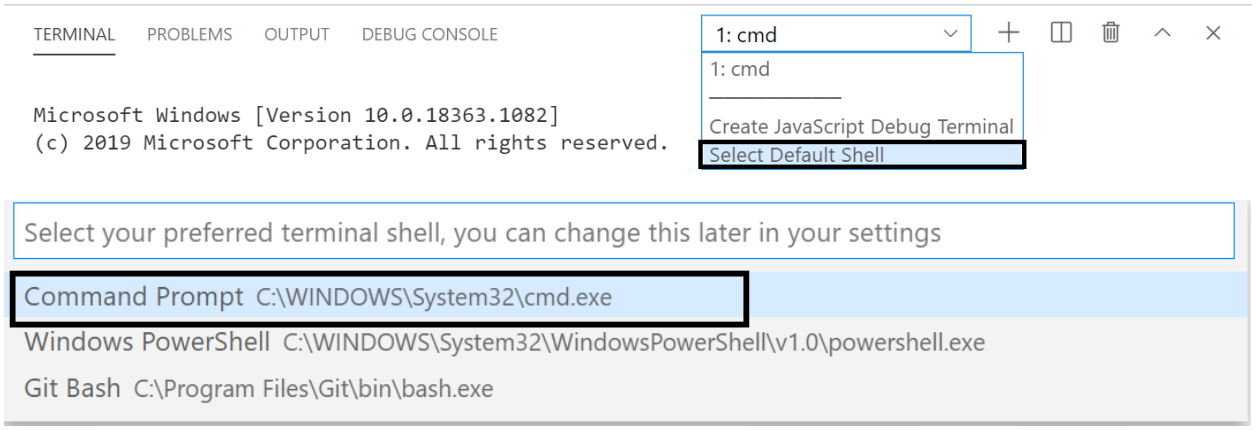


3. Open **VSCode-win32-x64-1.49.3** folder open the VS Code editor by clicking Code.exe

4. Open the folder **ng10-app** in visual studio code editor



5. Open New Terminal and Select Default Shell as **cmd**.exe and restart the Visual Studio code editor.



Run the Batch file **ng10path.bat** to set the Environment variables for the current session

```
\Environment\ng10-app>ng10path.bat
```

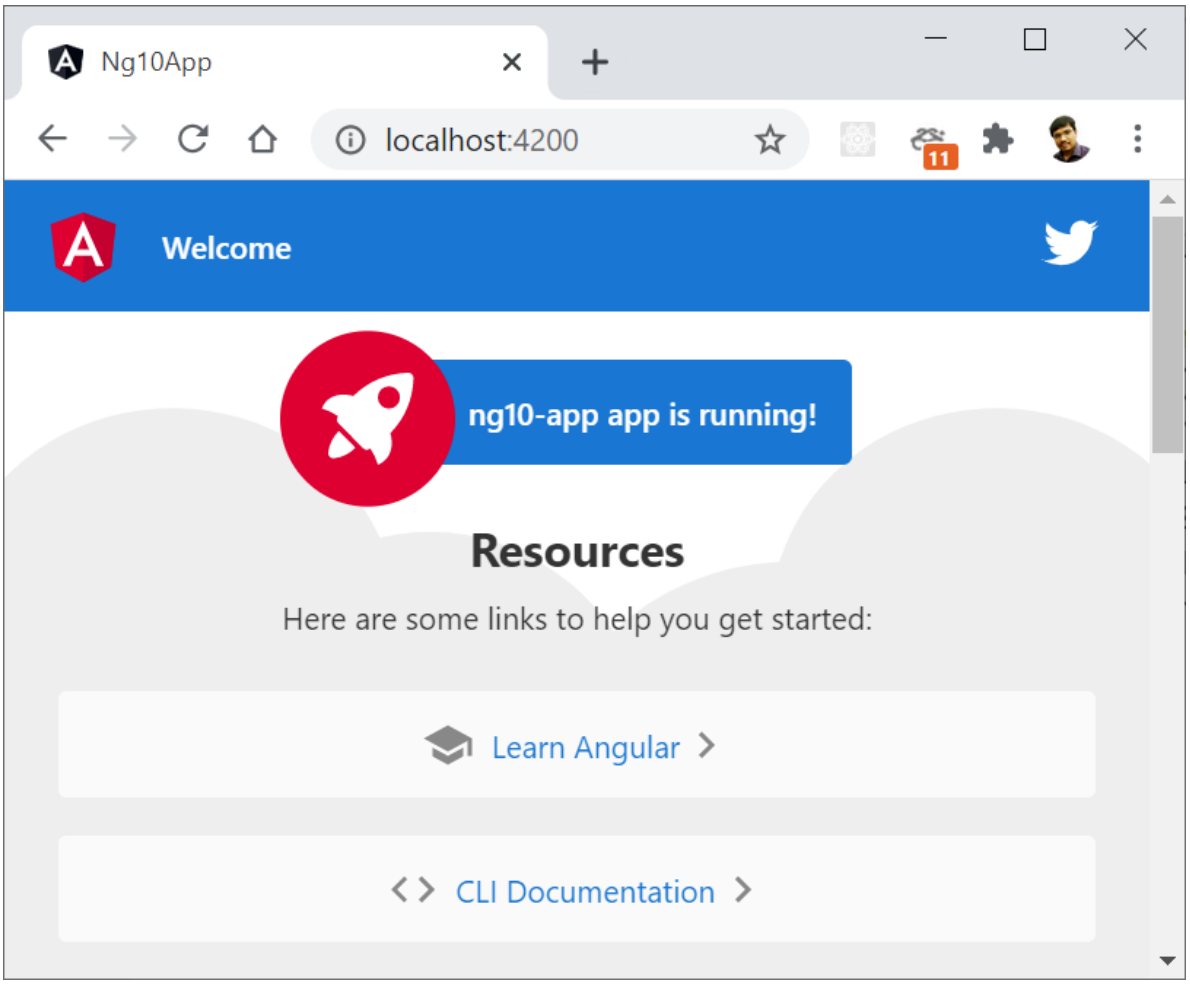
6. Verify node, npm and Angular cli version using the following command (node -v, npm -v, ng --version)

```
C:\Users\karmuthu\Desktop\Masterclass\Angular\Environment\ng10-app>node -v
v12.19.0

C:\Users\karmuthu\Desktop\Masterclass\Angular\Environment\ng10-app>npm -v
6.14.8

C:\Users\karmuthu\Desktop\Masterclass\Angular\Environment\ng10-app>ng --version

     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 10.1.7
Node: 12.19.0
OS: win32 x64
```

7. Run the command **ng serve** to build and run the angular application

```
C:\Users\karmuthu\Desktop\Masterclass\Angular\Environment\ng10-app>ng serve

chunk {main} main.js, main.js.map (main) 57 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 12.5 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.38 MB [initial] [rendered]
Date: 2020-10-18T08:15:52.511Z - Hash: a6e1166e891f85bb7df3 - Time: 16813ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://l
ocalhost:4200/ **
: Compiled successfully.
```

8. Open Google Chrome and type the URL **http://localhost:4200** and verify the results as shown below



Task 01: Adding Navigation Bar

Estimated Time : 30 mins

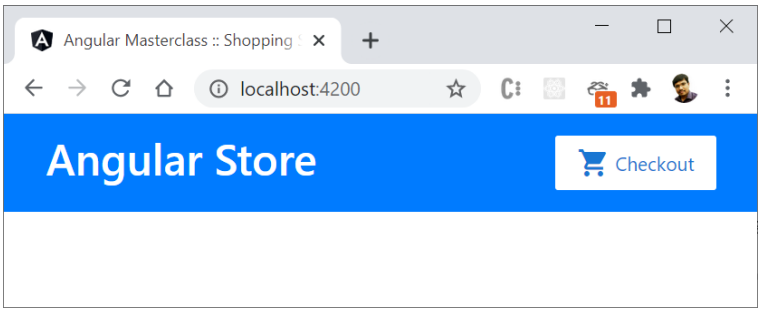
Task : In this task you will create an angular component named **NavBarComponent** , which serves as Top Navigational Bar for the shopping site with Heading and Checkout button.

Shared Artifacts : Use Angular Environment which has been created in the last task and use the files and the articles shared with you (**Task01-Sharables.zip**) to complete this task.

Learnings:

- How to use Interpolation
- Create component using Ng command and styling it using bootstrap.
- Using google icons in Angular application

Final Outcome:



Guided Steps :

a. In the project we have already installed bootstrap 4.5.3 as Save Dependency.
   - Include the bootstrap css in the angular.json file to include it in the angular build (Refer **How to Add Bootstrap to an Angular CLI project.pdf** from the Task01-Sharables folder)

b. To use the Google icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

c. Add the css code given in **styles.css** to the global styles file named styles.css in the src folder

d. Generate the component named nav-bar using ng generate command :

ng g component nav-bar --skipTests

e. In the **nav-bar.component.html** create a Navbar as by referring bootstrap documentation page

f. Create a property named **title** with the value **Angular Store** in **NavBarComponent.ts** and bind with h1 tag in **nav-bar.component.html**

g. Add the **< app-nav-bar>** tag inside the div element in **app.component.html** to render the component

Reference Links :

https://getbootstrap.com/docs/4.5/components/navbar/
https://material.io/resources/icons/?style=baseline

Task 02: Adding SideBar populated with menu items

Estimated Time : 30 mins

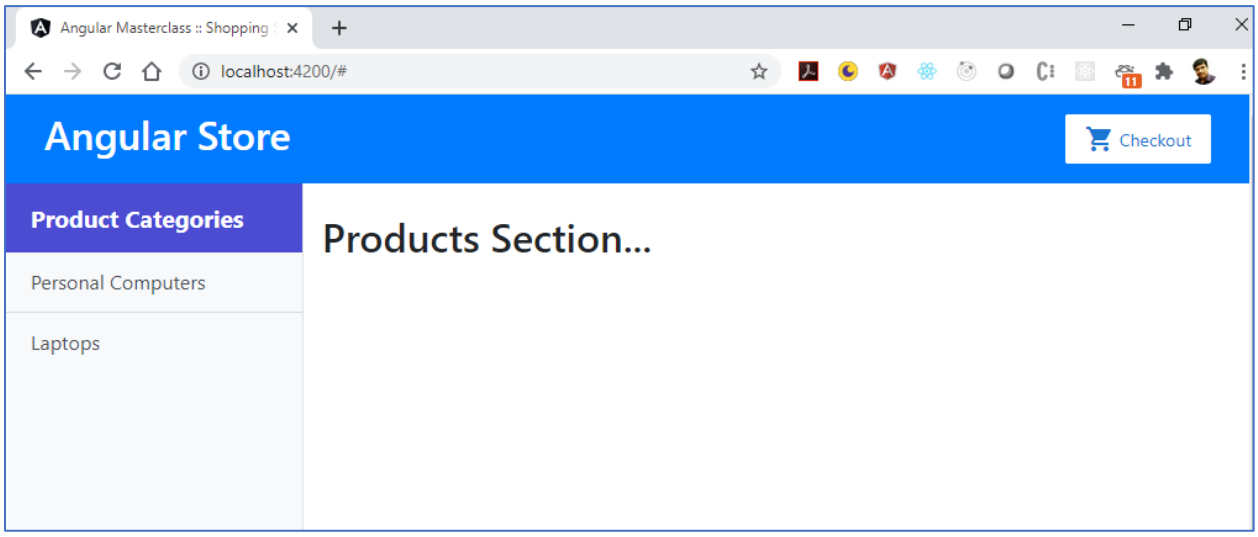
Task : In this task you will create an angular component named **SideBarComponent,** which serves as a container (Side-Bar) to hold the product category list as Menu Items

Shared Artifacts : Use src folder by extracting the ***Task02-Sharables.zip*** which contains the skeletal code and use the files shared in that to complete this task.

Learnings:

- How to use databinding & Angular structural directives
- Create angular components using the bootstrap templates

Final Outcome:



Guided Steps

a. Add the code for creating SideBar in **side-bar.component.css(sidebar.css in shared folder)** and **side-bar.component.html** by reffering the bootstrap side bar bootstrap free template **startbootstrap-simple-sidebar-gh-pages.zip** shared with you

b. Create **categories.ts** in app folder and export the categories shared in the **categories.txt** file

c. Create the category menu items in **SideBarComponent** using **NgFor** Structural directive by importing categories from **categories.ts** and creating property named categories in **side-bar.component.ts**

d. Add the approriate code in **app.component.html** to render the **SideBarComponent**

Task 03: Creating Product Gallery with images

Estimated Time : 30 mins

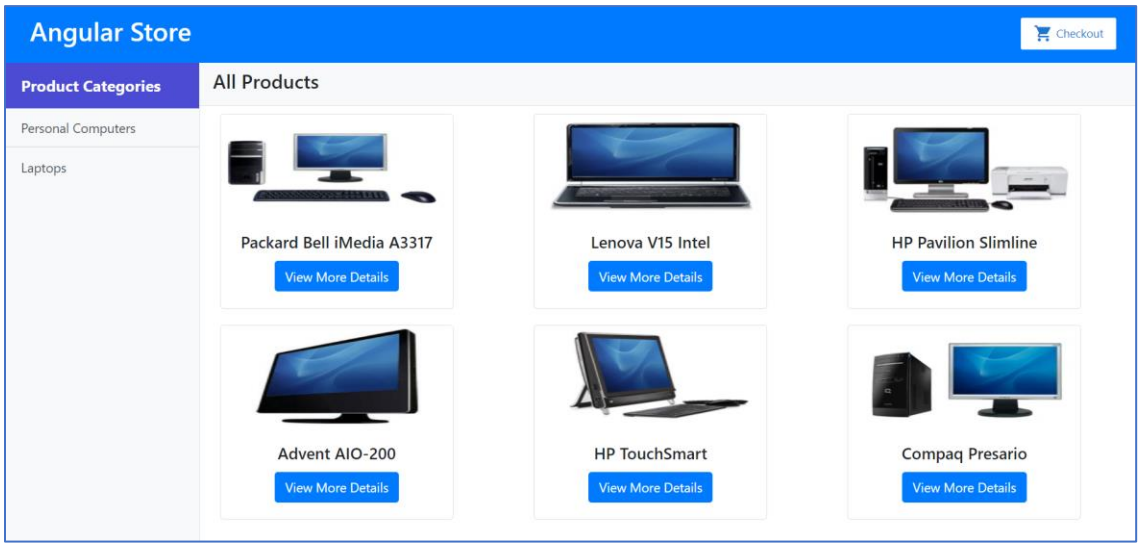
Task : In this task you will create an angular component named **ContentBarComponent** which populates a galley with product name and thumbnail images

Shared Artifacts : Use src folder by extracting the ***Task-03-starter.zip*** which contains the skeletal code and use the files & images shared in that to complete this task.

Learnings:

- How to use bootstrap card component to create gallery
- Add images to the angular project

Final Outcome:



Guided Steps:

a. Create folder named **images** under **assets** folder and add the images (product images) shared with you

b. Create **products.ts** in app folder and export the products shared in the **products.txt** file

c. Create the Product gallery in **ContentBarComponent** using Bootstrap Card component by importing products from **products.ts**

d. Add the approriate code in **app.component.html** to render the **ContentBarComponent**

References:

https://getbootstrap.com/docs/4.5/components/card/

Task 04: Creating and Nesting Product Component

Estimated Time : 30 mins

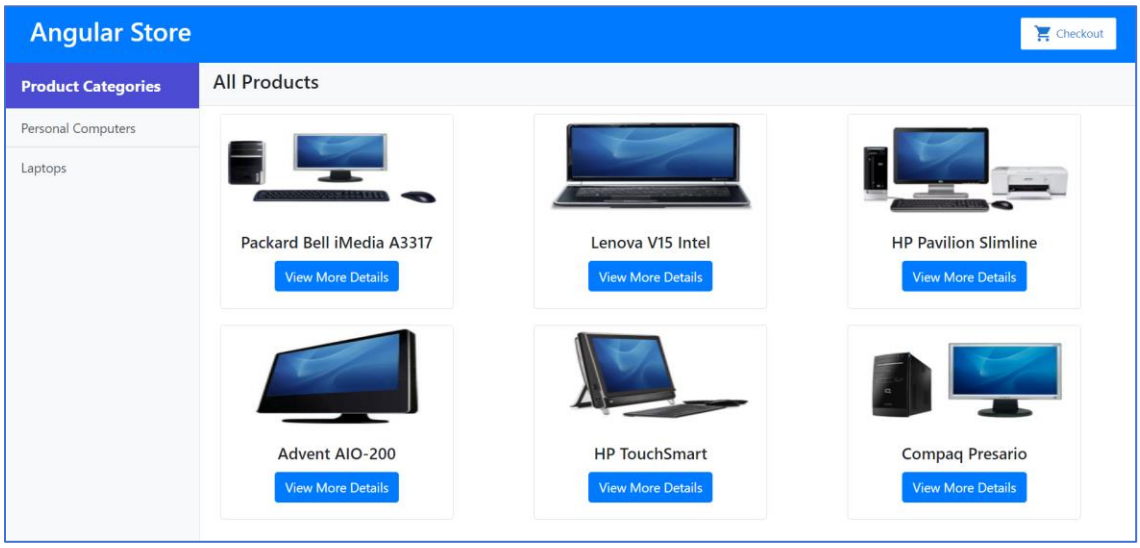
Task : In this task you will create an angular component named **ProductListInfoComponent** which recieves the product information from its parent component **ContentBarComponent.**

Shared Artifacts : Use src folder by extracting the ***Task-04-starter.zip*** which contains the skeletal code and use the files in that to complete this task.

Learnings:

- How to Nest Angular Components
- How to share data between the parent context and child directives or components.
- Create have a strongly typed model using interface

Final Outcome: (Similar as Task-03 Outcome)



Guided Steps:

a. Add the respective properties in the **category.ts** and **product.ts** and export the strongly typed models unders the **models** folder

b. Render the **ProductListInfoComponent** and pass the product details in **content-bar.component.html**

c. Add the approriate code in **product-list-info.component.ts** & **product-list-info.component.html** under **product-list-info** folder

Task 05: Adding Navigation

Estimated Time : 30 mins

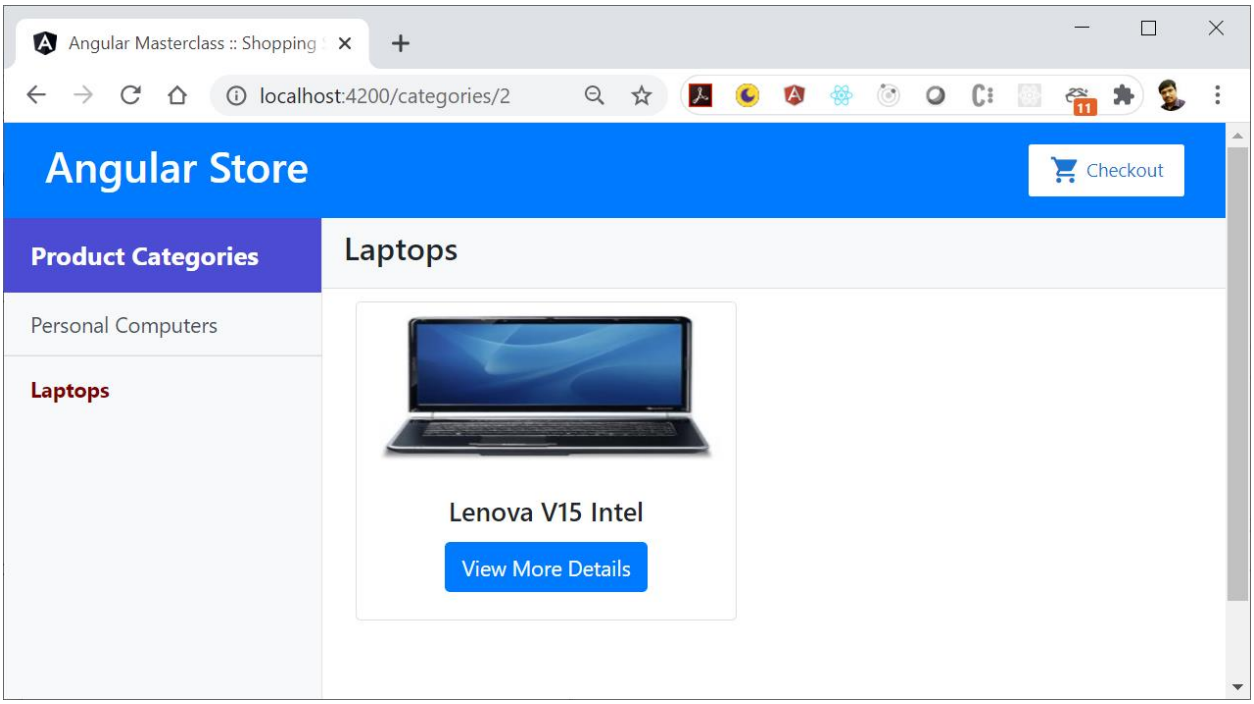
Task : In this task you will integrate Angular routing feature into your online store which helps us to display the appropriate product categories based on the Category Item clicked in the SideBarComponent**.**

Shared Artifacts : Use src folder by extracting the ***Task-05-starter.zip*** which contains the skeletal code and use the files in that to complete this task.

Learnings:

- How to Implement Angular's Routing Feature
- How to display components based on the browser's URL and your defined routes..
- How to navigate to a new view by clicking links on the page.
- How to add a placeholder that Angular dynamically fills based on the current router state.

Final Outcome:



Guided Steps:

a. In **app.routes.ts**, add a route for category details, with a path of **categories/:categoryId** and **ContentBarComponent** for the component, which displays products based on the categoryId. Also create a default route which displays all the products and export the routes.

b. In **app.module.ts** import the **RouterModule** and Specify the Routes by importing it from app.routes.ts

c. Add the Directives **RouterLink** for navigation and **RouterLinkActive** for higlighting the active Route in side-bar.component.html

Use the style given below to highlight the Active Route in **style.css**

```
.highlight{
  font-weight: bold;
  color:maroon;
}
```

d. Add the logic in **content-bar.component.ts** to get the categoryId Passed via Route and display the products accordingly when the ContentBarComponent Initialize.

Task 06: Display Product Information

Estimated Time : 60 mins

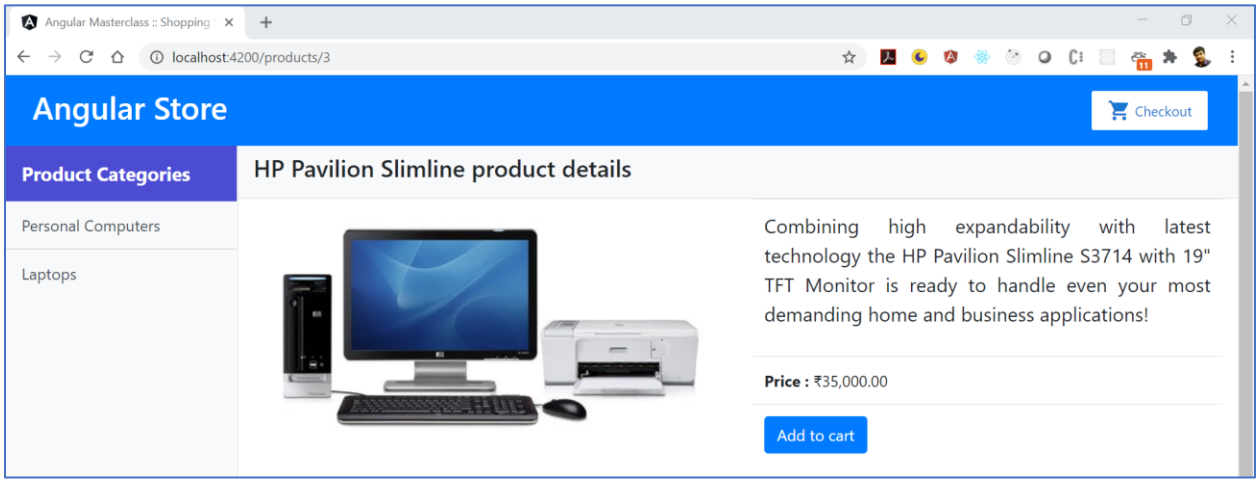
Task : In this task you will enchance the navigation when users click on View More Details button in the Gallery, the router should navigates them to the distinct URL for the product, swaps out the **ProductListInfoComponent** for the **ProductDetailsComponent**, and displays the product details.

Shared Artifacts : Use src folder by extracting the ***Task-06-starter.zip*** which contains the skeletal code and use the files in that to complete this task.

Learnings:

- How to Implement Angular's Routing Feature
- How to display components based on the browser's URL and your defined routes..
- How to navigate to a new view by clicking links on the page.
- How to add a placeholder that Angular dynamically fills based on the current router state.
- How to apply currency Pipe

Final Outcome:



Guided Steps:

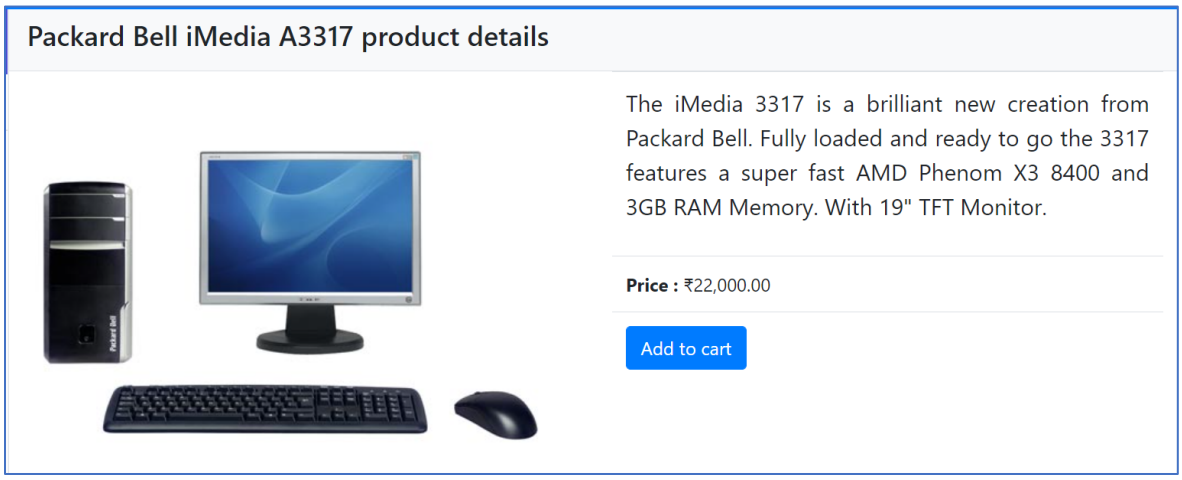
a. Generate the component named **product-details** using ng generate command :
ng g component product-details--skipTests

b. In **app.routes.ts**, add a route for product detail, with a path of **products/:productId** and **ProductDetailsComponent** for the component, which displays product information on the productId.

c. Add the **product-details.component.ts** and add the appropriate code to get the product id from the route and assign the product details in **product** property.

d. In **product-details.component.html** add the appropriate html to display the productName, description, proudct Image, price and Add to Cart Button as shown in Image below

e. Transform the price using Currency Pipe with India Rupees Symbol



Task 07: Manage Data in Cart using Angular Services

Estimated Time : 60 mins

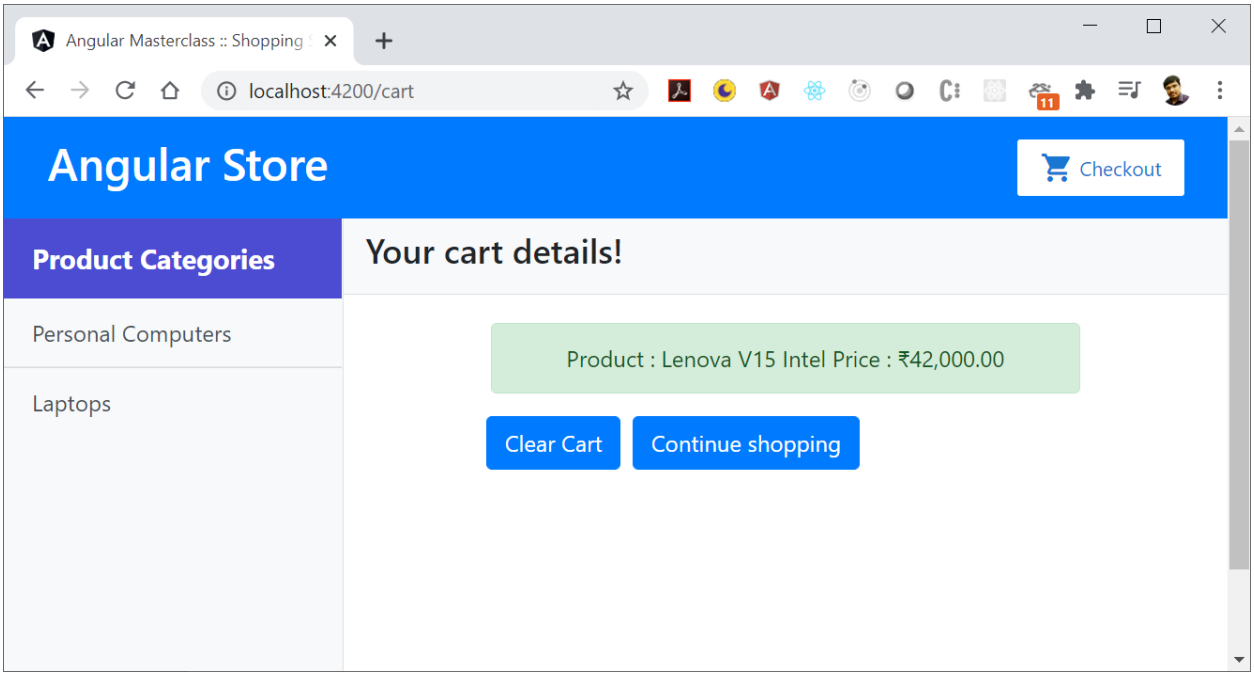
Task : In this task you will Create service named **CartService** which Add the current product into a list of products that a cart service manages(Add, Remove and Return all items) and also add a **CartComponent**, which displays the items in the cart by injecting the service created when the Checkout button is clicked.

Shared Artifacts : Use src folder by extracting the ***Task-07-starter.zip*** which contains the skeletal code and use the files in that to complete this task.

Learnings:

- How to create an Angular service which is an instance of a class that you can make the business logic created in that available to any part of your application using Angular's dependency injection system

Final Outcome:



Guided Steps:

a. Generate the service named **CartService** using ng generate command :
   ng g service services/cart --skip-tests

b. Create properties named **items** in **CartService** which holds collection of products and create the following three methods.
   i. addToCart(product) : Add products to the **items** collection
   ii. getItems() : return all the items

iii. clearCart() : clear all the items

c. Create the component named **CartComponent** and inject the **CartService.**

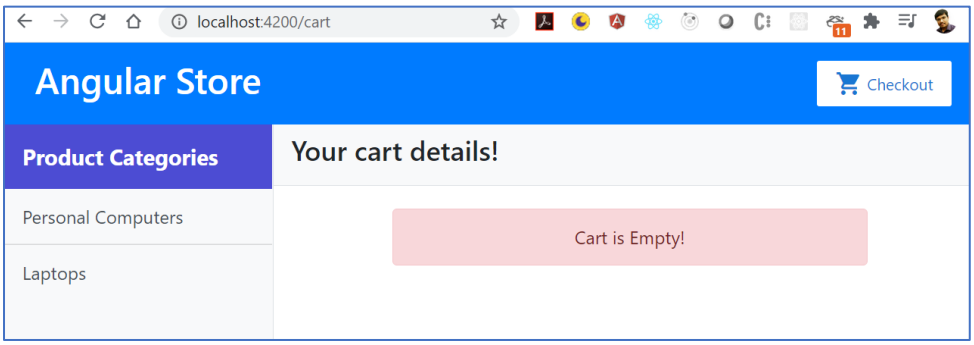
d. In app.routes.ts, add a route for cart, with a path of **/cart** and **CartComponent**.

e. In **nav-bar.component.html** add the RouterLink for the Checkout button which navigates to **CartComponent**

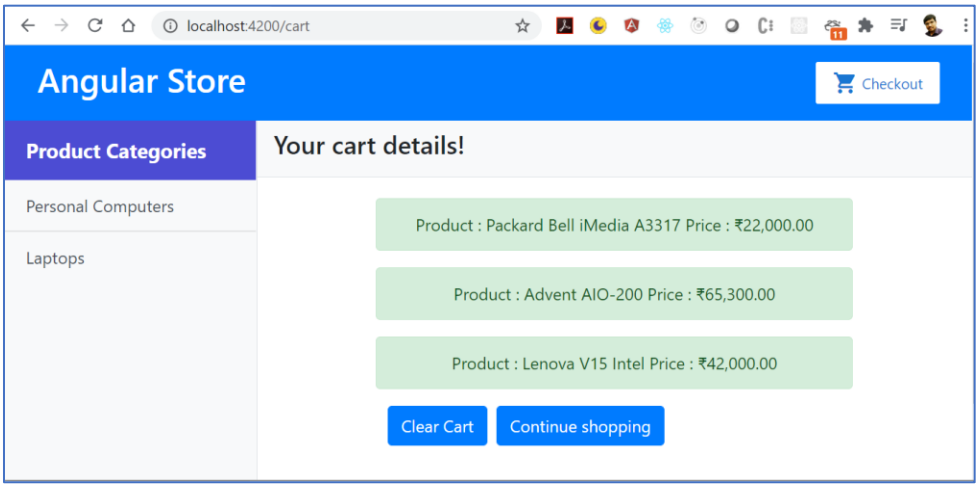
f. Initialize the **items:Array<Product>** with the getItems() from the CartService when the component intializes.

g. In the View Part **cart.component.html** create the details as given below

Display the message "**Cart is Empty!**" if no items added in cart



If items have been added to cart, Display the Product name with price followed with 2 buttons **ClearCart(Clears all the items in Cart)** and **Continue Shopping (Navigate to Default Route)**



Task 08: Form-based checkout feature

Estimated Time : 60 mins

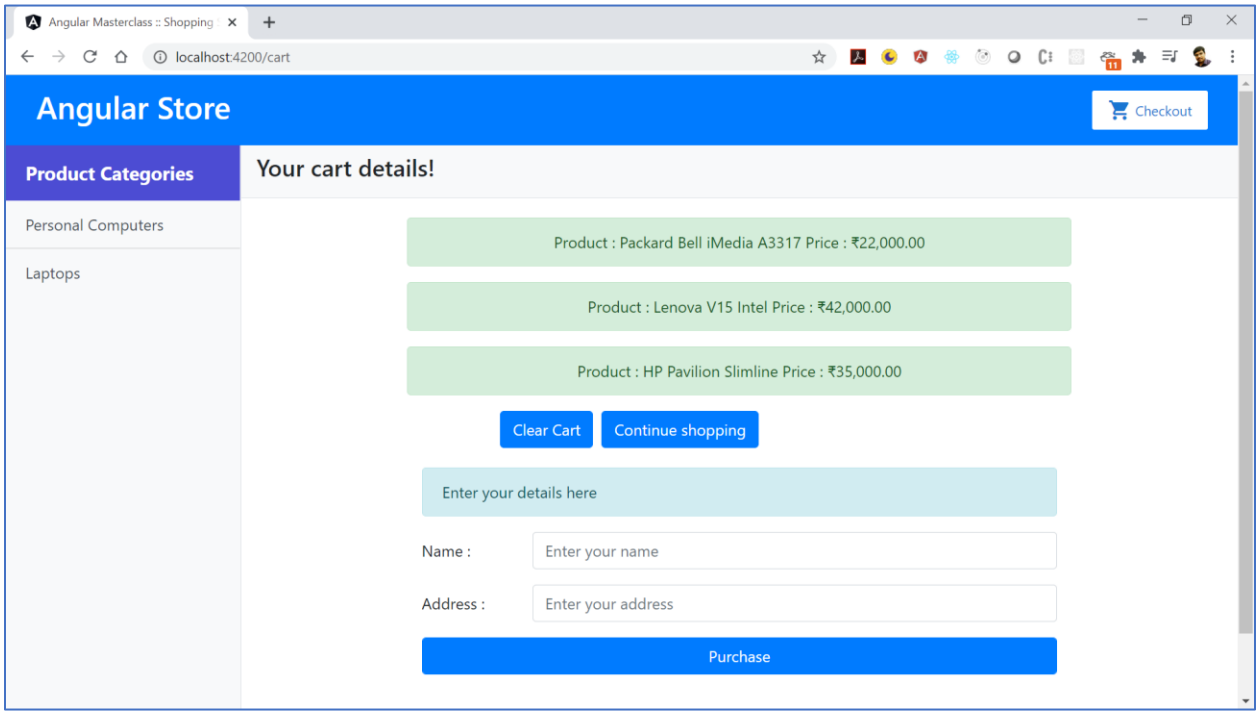
Task : In this task you will create a form-based checkout feature to collect user information as part of checkout.

Shared Artifacts : Use src folder by extracting the ***Task-08-starter.zip*** which contains the skeletal code and use the files in that to complete this task.

Learnings:

- How to use Angular Reactive Form which works with the object that live in components to store and Manage forms and the visualization of the form that lives in the template.

Final Outcome:



Guided Steps:

a. Open **cart.component.ts** import and Inject the **FormBuilder** service

b. Define the **checkoutForm** property to store the form model

c. To gather the user's name and address, set the **checkoutForm** property with a form model containing name and address fields, using the FormBuilder group() method.

d. In **cart.component.ts**, define an **onSubmit**() method to process the form.

   i. In the console screen print the User Details and Product Items added to the cart.

ii. Use the CartService clearCart() method to empty the cart items and reset the form after its submission.

e. Open **cart.component.html,** At the bottom of the template, add an bootstrap form to capture user information.

   i. Use a formGroup property binding to bind the checkoutForm to the form tag in the template. Also include a "**Purchase**" button to submit the form.

References:

https://getbootstrap.com/docs/4.5/components/forms/

Task 09: Creating Custom Element & Building Angular Project for Production

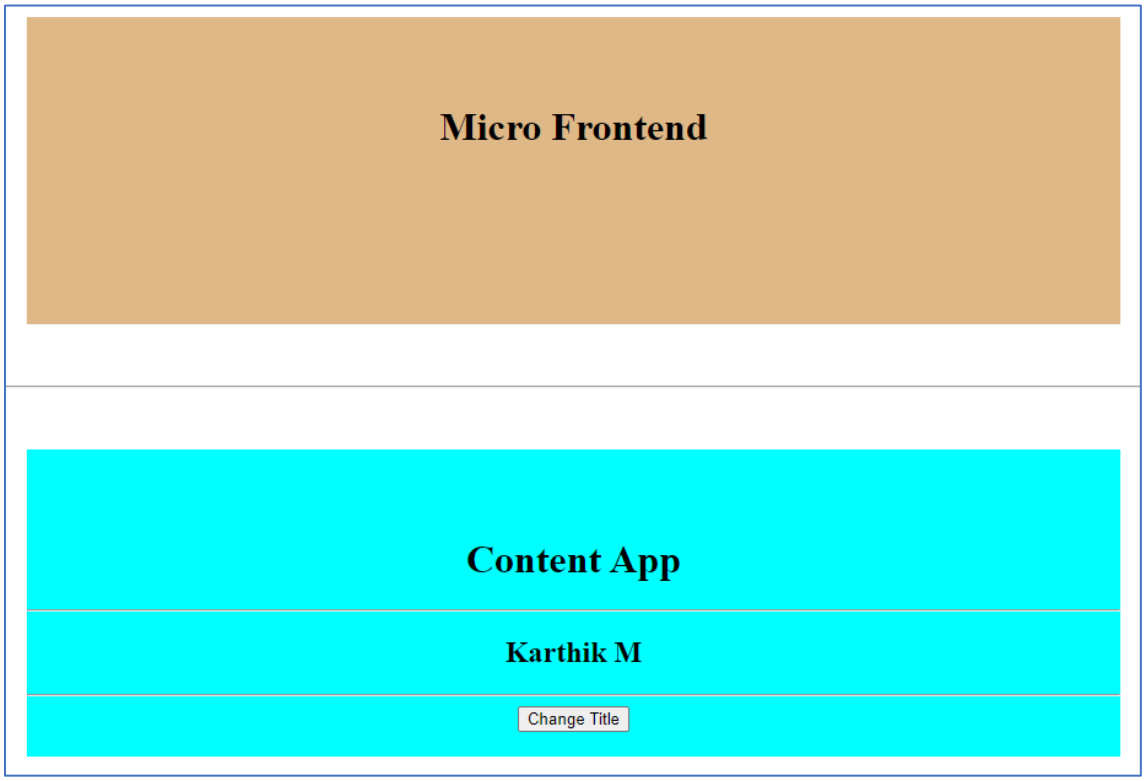
Estimated Time : 120 mins

Task : In this task you will create a Custom Element and Build the Angular Project for Production through Step-By-Step Instructions

Learnings:

- How to create and consume web components using Angular Elements
- Build Angular Project for Production
- Understand Micro Frontend Architecture

Final Outcome: Main Application shows the customElement which is a outcome of 2 Angular Project



Step by Step Instruction

Step 1: Install ***Angular CLI 10*** globally

```
npm i -g @angular/cli
```

Step 2: Create the Angular workspace with out application

```
ng new mf-workspace --createApplication="false"
```

Step 3: Generate Angular application named ***headerApp*** and ***contentApp***

```
ng generate application headerApp
ng generate application contentApp
```

Step 4: Add **@angular/elements**

```
ng add @angular/elements
```

Step 5: Add the following code in ***app.module.ts*** in ***headerApp***

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, Injector } from '@angular/core';
import { createCustomElement } from "@angular/elements";
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap:[],
  entryComponents:[
    AppComponent
  ]
})
export class AppModule {

  constructor(private injector:Injector){

  }

  ngDoBootstrap(){
    const headerApp =
createCustomElement(AppComponent,{injector:this.injector});
    customElements.define('header-app',headerApp);
  }
}
```

Step 6: Add the following code in ***app.module.ts*** in ***contentApp***

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, Injector } from '@angular/core';
import { createCustomElement } from "@angular/elements";
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
```

```
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap:[],
  entryComponents:[
    AppComponent
  ]
})
export class AppModule {

  constructor(private injector:Injector){

  }

  ngDoBootstrap(){
    const contentApp =
createCustomElement(AppComponent,{injector:this.injector});
    customElements.define('content-app',contentApp);
  }
```

Step 7: Extend the Angular CLI's default build behavior using ***ngx-build-plus*** *by adding it in* ***headerApp*** *and* ***contentApp*** *with the following commands*

Note*: Change the default Porject in angular.json to* ***"defaultProject": "headerApp"*** *to add* ***ngx-build-plus*** *in* ***headerApp*** *project similarly repeat the same while adding in* ***contentApp*** ***"defaultProject": "contentApp"***

```
ng add ngx-build-plus

ng g ngx-build-plus:wc-polyfill (Adds webcomponent polyfills to your app)

ng g ngx-build-plus:externals (Updates your app to use webpack externals)
```

Step 8: Add the following code snippets in **headerApp** Project

```
headerApp/src/styles.css
.header-style{
    background-color:burlywood;
    height:200px;
    text-align:center;
    padding-top: 50px;
    margin:50px;
}

headerApp/src/app/app.component.ts

import { Component, ChangeDetectorRef, ChangeDetectionStrategy,
ViewEncapsulation } from '@angular/core';

@Component({
```

```
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Micro Frontend';

}
```

headerApp/src/app/app.component.html

```
<div class="header-style">
  <h1>{{ title }}</h1>
</div>
```

headerApp/src/index.html

```
<body>
  <header-app></header-app> <!--instead of <app-root></app-root> -->
</body>
```

Step 9: Add the following code snippets in **contentApp** Project

contentApp/src/styles.css

```
.content-style{
    background-color:burlywood;
    height:200px;
    text-align:center;
    padding-top: 50px;
    margin:50px;
}
```

contentApp/src/app/app.component.ts

```
import { Component} from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

}
```

contentApp/src/app/app.component.html

```
<div class="content-style">
    <h1>Content App</h1>
</div>
```

contentApp/src/index.html

```
<body>
  <content-app></content-app> <!--instead of <app-root></app-root> -->
</body>
```

Note: By this stage we can build the indicidual projects by typing the following command

```
npm run build:headerApp:externals
npm run build:contentApp:externals
```

Step 10: Inorder to bundle the build files(javaScript files) Install the following packages as Developer Dependencies

```
npm install -D fs-extra
npm install -D concat
```

Step 11: create a file named **build-elements.js** in the **src** folder to build and create the custom elements.

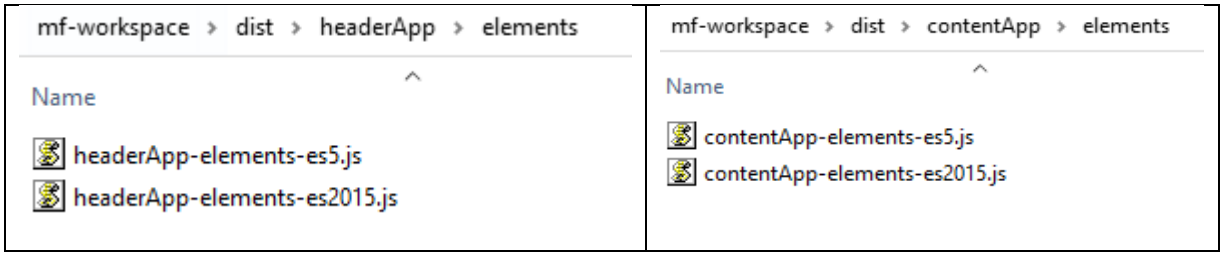
```
const fs = require('fs-extra');
const concat = require('concat');

(async function build(){
    const prgName = process.argv.slice(2)[0];
    if(prgName == '' || prgName == undefined){
        console.log('Provide Project name as argument');
        return false;
    }else{
        const files_es2015 = [
            './dist/' + prgName + '/polyfill-webcomp-es5.js',
            './dist/' + prgName + '/polyfill-webcomp.js',
            './dist/' + prgName + '/polyfills.js',
            './dist/' + prgName + '/scripts.js',
            './dist/' + prgName + '/main.js'
        ]
        await fs.ensureDir('./dist/' + prgName + '/elements');
        await concat(files_es2015, './dist/' + prgName + '/elements/' +
prgName + '-elements-es2015.js');

        const files_es5 = [
            './dist/' + prgName + '/polyfill-webcomp-es5.js',
            './dist/' + prgName + '/polyfill-webcomp.js',
            './dist/' + prgName + '/polyfills.js',
            './dist/' + prgName + '/scripts.js',
            './dist/' + prgName + '/main.js'
        ]
        await fs.ensureDir('./dist/' + prgName + '/elements');
```

```
        await concat(files_es5, './dist/' + prgName + '/elements/' + prgName
+ '-elements-es5.js');

        console.log('Done generating bundles for '+ prgName);
    }
})();
```

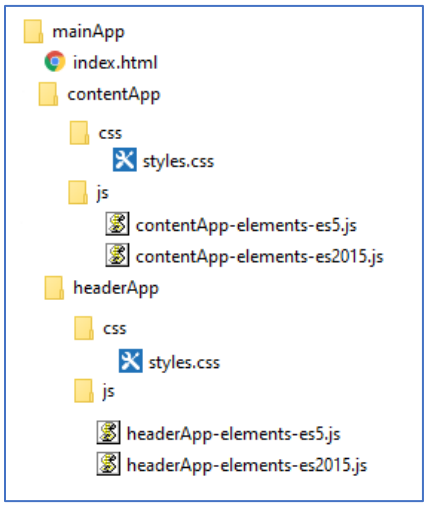
Step 12: Change the autogenerated build command to create single build file by avoiding hashing names and building it using build-elements.js as given below.

```
"build:headerApp:externals": "ng build --extra-webpack-config
projects/header-app/webpack.externals.js --prod --project headerApp --single-
bundle --output-hashing none && node build-elements.js headerApp",

"build:contentApp:externals": "ng build --extra-webpack-config
projects/content-app/webpack.externals.js --prod --project contentApp --
single-bundle --output-hashing none && node build-elements.js contentApp"
```

Step 13: Run the build again for every Custom Element project which produce the following files under dist\headerApp\elements & dist\contentApp\elements folder .



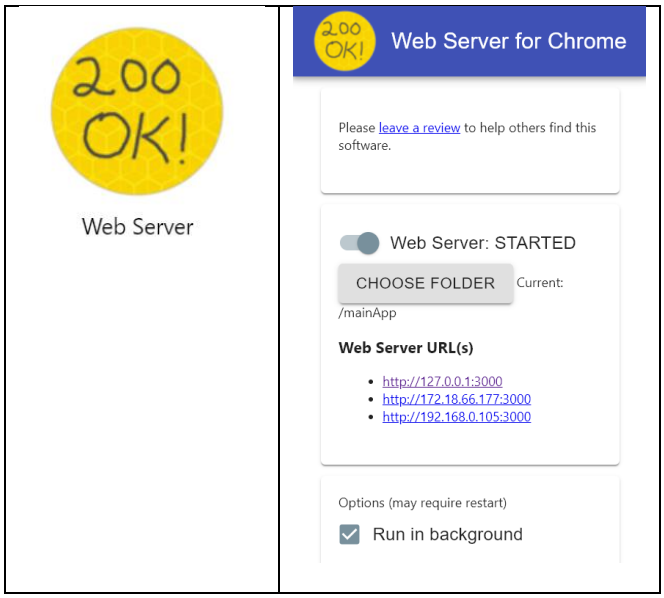
Step 14: Create a folder named mainApp with the following Folder Structure and copy the js & css files.



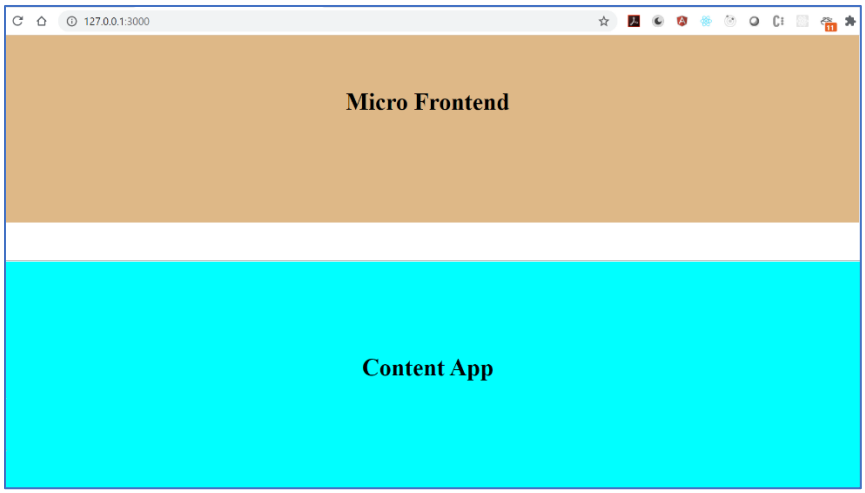
Step 14: Add the following content in **index.html**

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MainApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="./headerApp/css/styles.css">
  <link rel="stylesheet" href="./contentApp/css/styles.css">
<body>
  <div>
  <header-app></header-app>
  </div>
  <hr>
  <div>
  <content-app></content-app>
  </div>
  <script src="./headerApp/js/headerApp-elements-es2015.js"></script>
  <script src="./contentApp/js/contentApp-elements-es2015.js"></script>
</html>
```

Step 15: Serve the contents in mainApp folder using any Webserver (like webserver for chrome – Chrome Plugin)



Step 16: headerApp and contentApp custom Elements rendered in the mainApp



Step 17 : We can pass details in to the component using Property Binding. Add the following code Snippets and rebuild the application(**contentApp**)

```
content-app/src/app/app.component.ts

export class AppComponent {
  @Input() public authorName:string;
}

content-app/src/app/app.component.html

<div class="content-style">
  <h1>Content App</h1>
  <hr>
  <h2>{{ authorName }}</h2>
</div>

content-app/src/index.html

<body>
  <content-app author-name="Karthik"></content-app>
</body>
```

Step 18 : Similarly we can communicate between custom elements through custom events. Add the following code snippets and rebuild the applications(**contentApp** & **headerApp**)

```
content-app/src/app/app.component.ts

export class AppComponent {
  @Input() public authorName:string;

  changeTitle(){
    const data = {
```

```
      title:'Changed Title'
    }
    const event = new CustomEvent('change_title_event',{detail:data});
    window.dispatchEvent(event);
  }
}
```

content-app/src/app/app.component.html

```
<div class="content-style">
  <h1>Content App</h1>
  <hr>
  <h2>{{ authorName }}</h2>
  <hr>
  <button (click)="changeTitle()">Change Title</button>
</div>
```

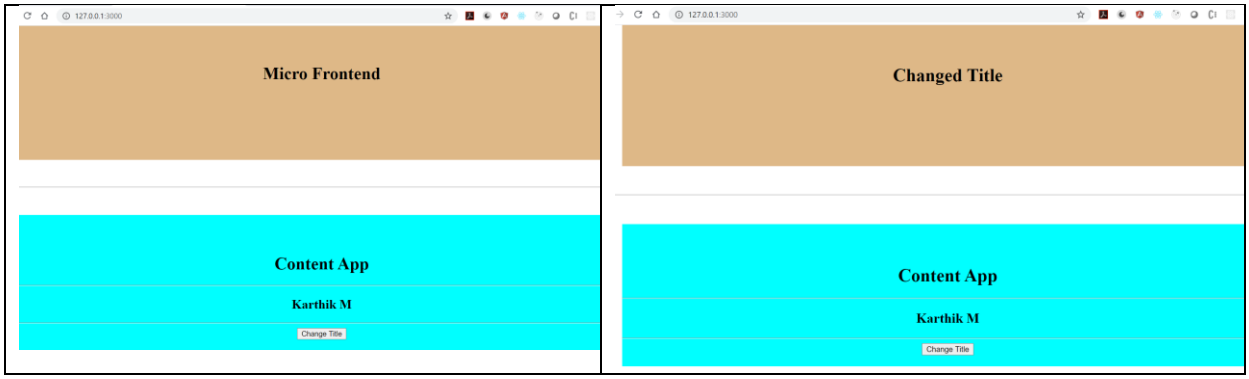
content-app/src/index.html

```
<body>
  <content-app author-name="Karthik"></content-app>
</body>
```

header-app/src/app/app.component.ts

```
export class AppComponent {
  title = 'Micro Frontend';

  constructor(private cd:ChangeDetectorRef){

  }

  ngOnInit(){

window.addEventListener('change_title_event',this.changeHeaderTitle.bind(this
),true);

  }

  changeHeaderTitle(eventData){
    this.title = eventData.detail.title;
    this.cd.detectChanges();
  }

  ngOnDestroy(){

window.removeEventListener('change_title_event',this.changeHeaderTitle,true);
  }
}
```

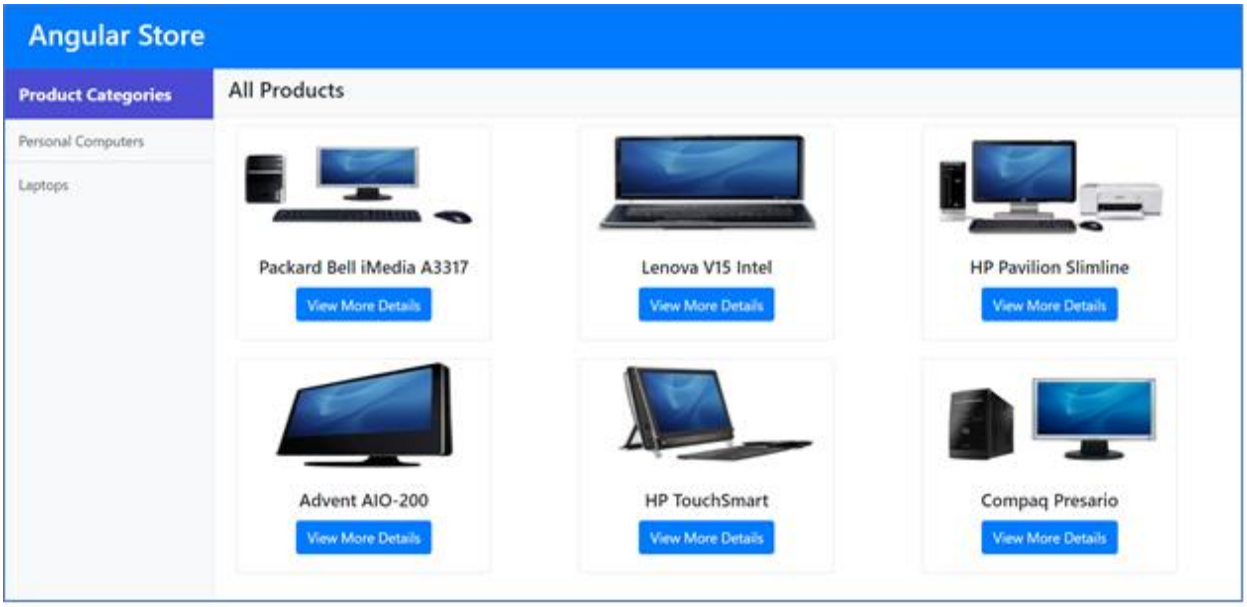
Final Outcome



Task 10: Creating Custom Element,Unit Testing & Building Angular Project for Production

Estimated Time : 240 mins

Task : In this task you need to design the following screen by as 3 Custom Elements (headerApp,linkApp & productApp) and write Unit Test cases for linkapp and productApp with All the best practices like Angular linting and also use Angular Services and Routing whereever applicable.



About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2017 global revenues of EUR 12.8 billion.

Learn more about us at www.capgemini.com.