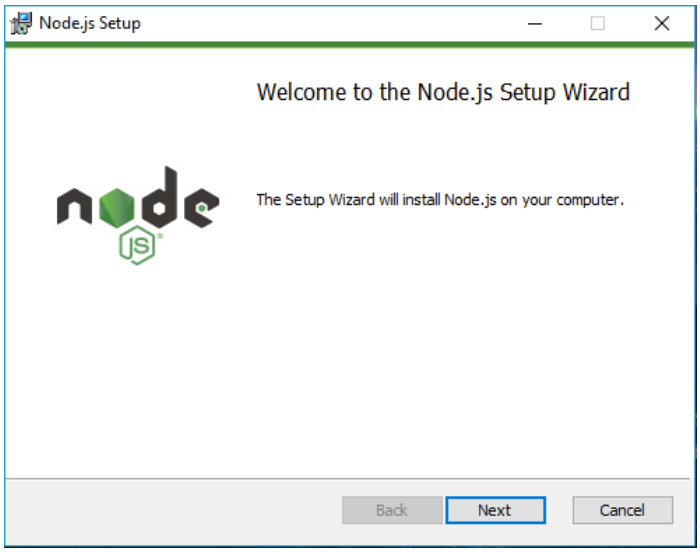
1.a Course Name: Angular JS
Module Name: Angular Application Setup
Observe the link http://localhost:4200/welcome on which the mCart application is running. Perform the below activities to understand the features of the application.
Program:
Step-2: Running the Node.js installer.
Now you need to install the node.js installer on your PC. You need to follow the following steps for the Node.js to be installed:-

- Double click on the .msi installer.

  The Node.js Setup wizard will open.

- Welcome To Node.js Setup Wizard.

  Select "Next"



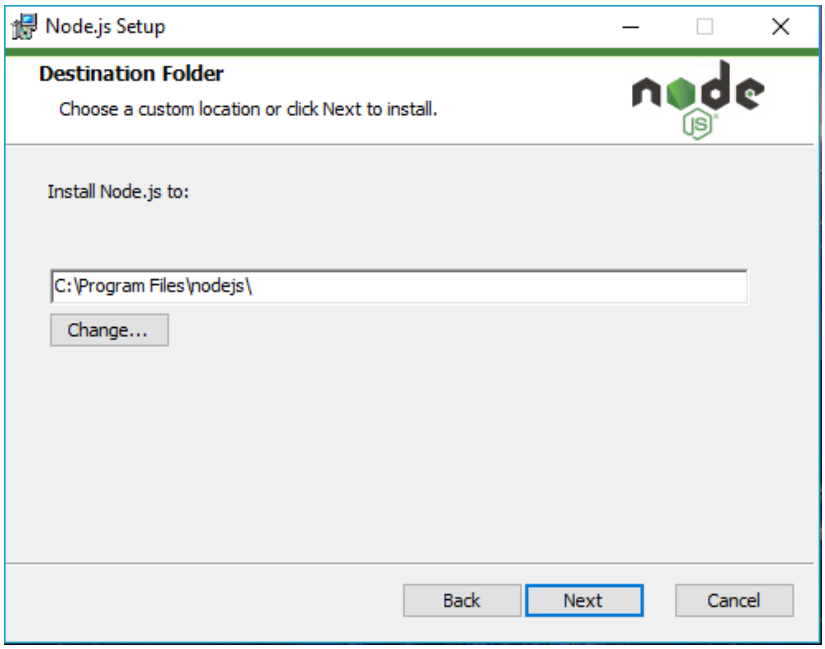
- After clicking "Next", End-User License Agreement (EULA) will open.

  Check "I accept the terms in the License Agreement"

  Select "Next"

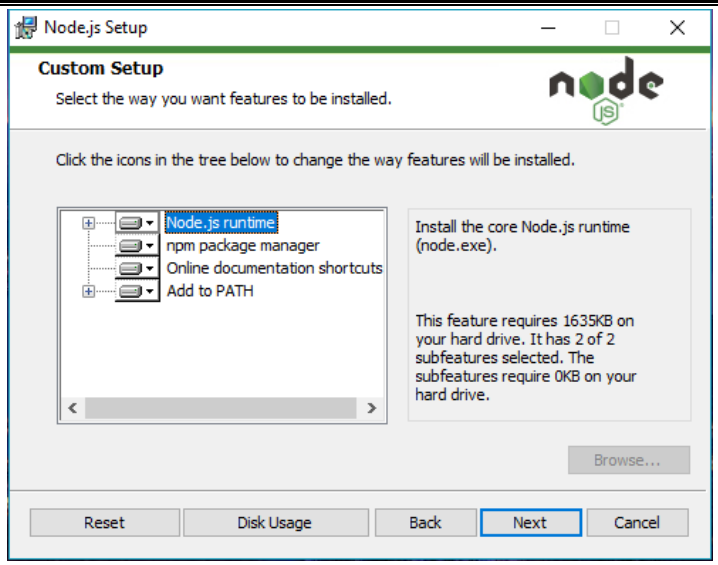
- Destination Folder

Set the Destination Folder where you want to install Node.js & Select "Next"



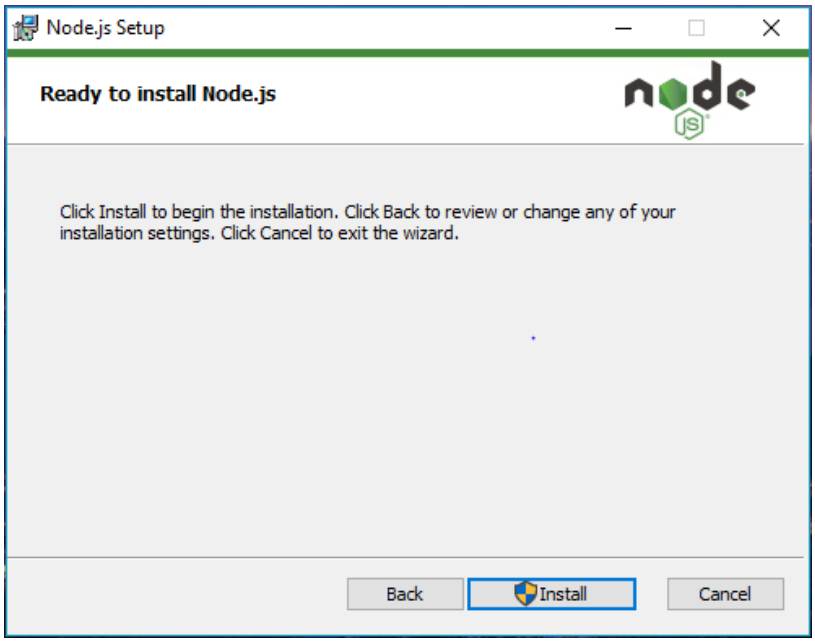
- Custom Setup

Select "Next"



- Ready to Install Node.js.

  Select "Install"

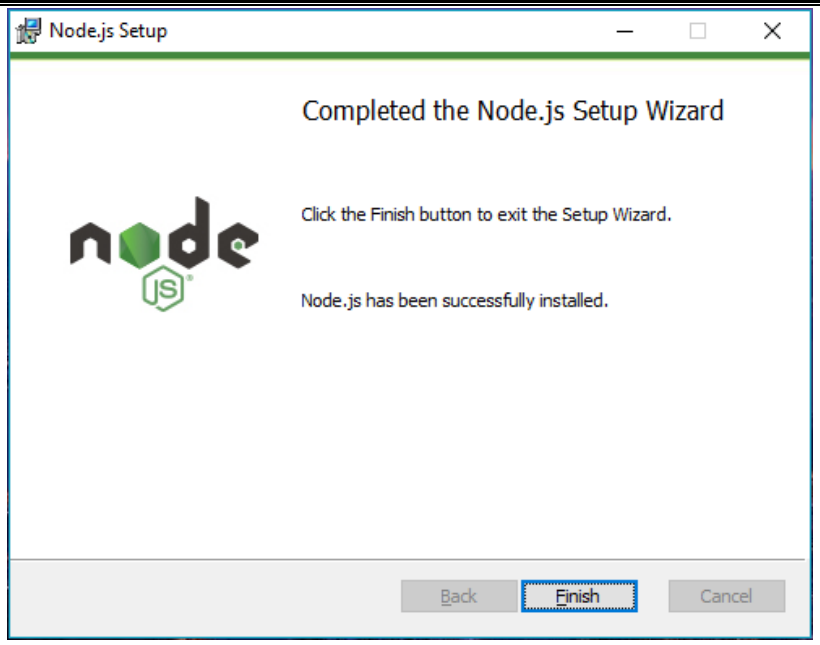


NOTE :
A prompt saying – "This step requires administrative privileges" will appear.

Authenticate the prompt as an **"Administrator"**

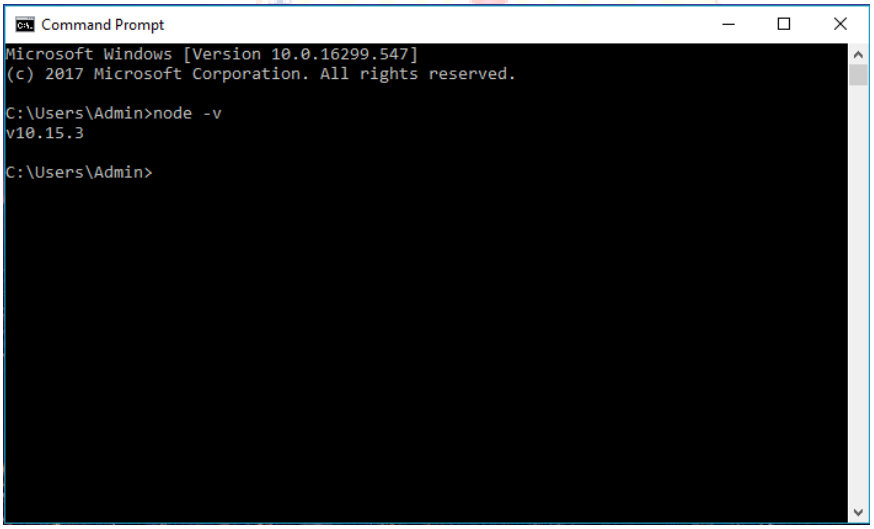
- Installing Node.js.
  Do not close or cancel the installer until the install is complete
- Complete the Node.js Setup Wizard.

  Click "Finish"



Step 3: Verify that Node.js was properly installed or not.
To check that node.js was completely installed on your system or not, you can run the following command in your command prompt or Windows Powershell and test it:-

C:\Users\Admin> node -v



If node.js was completely installed on your system, the command prompt will print the version of the node.js installed.

Step 4: Updating the Local npm version.
The final step in node.js installed is the updation of your local npm version(if required) – the package manager that comes bundled with Node.js.

You can run the following command, to quickly update the npm

npm install npm –global // Updates the 'CLI' client

C:\WINDOWS\system32\cmd.exe

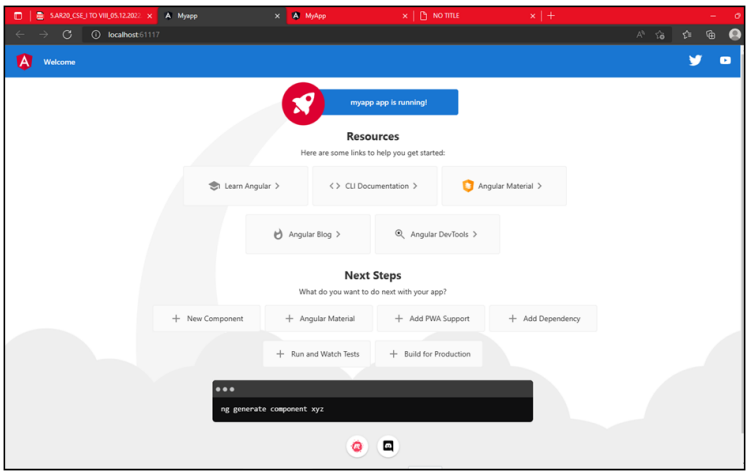
```
Microsoft Windows [Version 10.0.19045.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>node -v
v18.13.0

C:\Users\admin>npm -v
8.19.3
```

```
C:\Users\admin>npm install -g @angular/cli
npm WARN deprecated @npmcli/move-file@2.0.1: This functionality has been moved to @npmcli/fs

added 1 package, changed 224 packages, and audited 226 packages in 49s

27 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
C:\Users\admin>ng v

Angular CLI: 15.1.5
Node: 18.13.0
Package Manager: npm 8.19.3
OS: win32 x64

Angular:
...

Package                      Version
---------------------------------------------------------
@angular-devkit/architect    0.1501.5 (cli-only)
@angular-devkit/core         15.1.5 (cli-only)
@angular-devkit/schematics   15.1.5 (cli-only)
@schematics/angular          15.1.5 (cli-only)
```

```
C:\Users\admin>ng new mynewapp
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE mynewapp/angular.json (2710 bytes)
CREATE mynewapp/package.json (1039 bytes)
CREATE mynewapp/README.md (1062 bytes)
CREATE mynewapp/tsconfig.json (901 bytes)
CREATE mynewapp/.editorconfig (274 bytes)
CREATE mynewapp/.gitignore (548 bytes)
CREATE mynewapp/tsconfig.app.json (263 bytes)
CREATE mynewapp/tsconfig.spec.json (273 bytes)
CREATE mynewapp/.vscode/extensions.json (130 bytes)
CREATE mynewapp/.vscode/launch.json (474 bytes)
CREATE mynewapp/.vscode/tasks.json (938 bytes)
CREATE mynewapp/src/favicon.ico (948 bytes)
CREATE mynewapp/src/index.html (294 bytes)
CREATE mynewapp/src/main.ts (214 bytes)
```

```
C:\Users\admin\mynewapp>ng serve
? Port 4200 is already in use.
Would you like to use a different port? Yes
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   2.04 MB |
polyfills.js          | polyfills     | 314.27 kB |
styles.css, styles.js | styles        | 209.40 kB |
main.js               | main          |  10.11 kB |
runtime.js            | runtime       |   6.51 kB |

                      | Initial Total |   2.57 MB
```

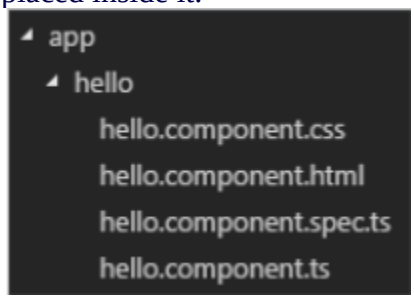


1.b Course Name: Angular JS
Module Name: Components and Modules
Create a new component called hello and render Hello Angular on the page.
Program:

1. In the same **MyApp** application created earlier, create a new component called hello using the following CLI command

```
1. D:\MyApp> ng generate component hello
```

2. This command will create a new folder with the name hello with the following files placed inside it.



3. Open **hello.component.ts** file and create a property called courseName of type string and initialize it to "Angular" as shown below in Line number 9

```
1.  import { Component, OnInit } from '@angular/core';
2.
3.  @Component({
4.    selector: 'app-hello',
5.    templateUrl: './hello.component.html',
6.    styleUrls: ['./hello.component.css']
7.  })
8.  export class HelloComponent implements OnInit {
9.    courseName: string = "Angular";
10.
11.   constructor() { }
12.
13.   ngOnInit() {
14.   }
15.

16. }
```

4. Open **hello.component.html** and display the courseName as shown below in Line 2

```
1. <p>
2.   Hello {{ courseName }}

3. </p>
```

5. Open **hello.component.css** and add the following styles for the paragraph element

```
1. p {
2.     color:blue;
3.     font-size:20px;

4. }
```

6. Open **app.module.ts** file and add HelloComponent to bootstrap property as shown below in Line 11 to load it for execution

```
1.  import { NgModule } from '@angular/core';
2.  import { BrowserModule } from '@angular/platform-browser';
3.
4.  import { AppRoutingModule } from './app-routing.module';
5.  import { AppComponent } from './app.component';
6.  import { HelloComponent } from './hello/hello.component';
7.
8.  @NgModule({
9.    imports: [BrowserModule,AppRoutingModule],
10.   declarations: [AppComponent, HelloComponent],
11.   providers: [],
12.   bootstrap: [HelloComponent]
13. })

14. export class AppModule { }
```

1. Open **index.html** and load the hello component by using its selector name i.e., app-hello as shown below in Line 11

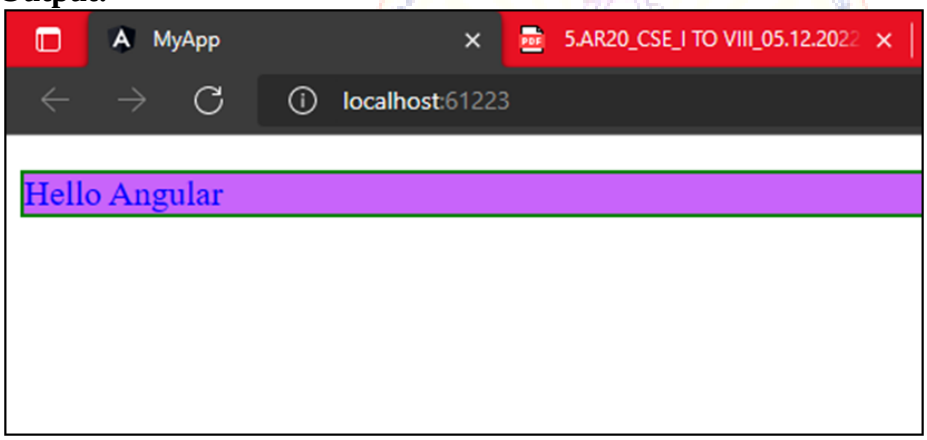
```
1. <!doctype html>
2. <html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <title>MyApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-hello></app-hello>
</body>

</html>
```

8. Now run the application by giving the following command

```
D:\MyApp>ng serve --open
```

Output:



1.c Course Name: Angular JS
Module Name: Elements of Template
Add an event to the hello component template and when it is clicked, it should change the courseName.
Program:

1. Open **hello.component.ts**, add a method called changeName() as shown below in Line 12-14. Also, use external template hello.component.html.

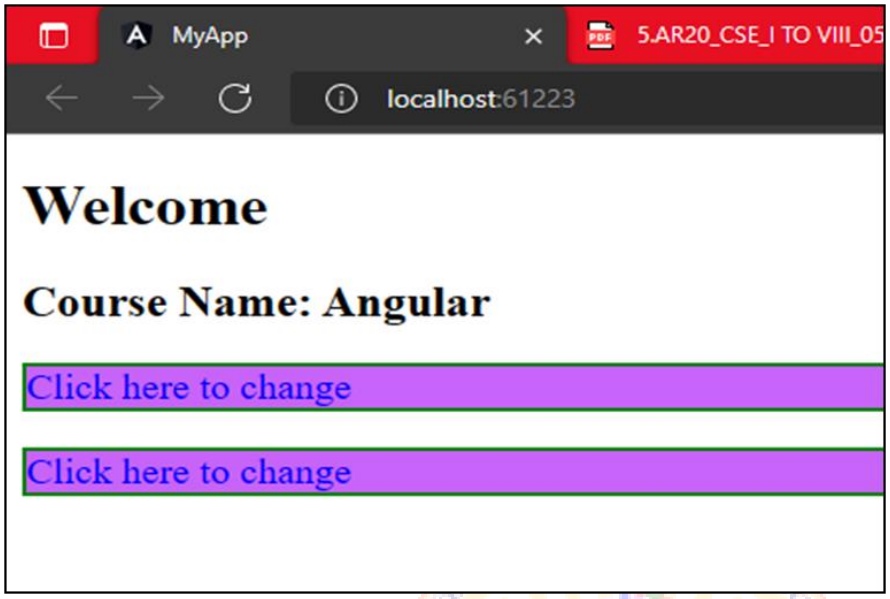
```
1.  import { Component, OnInit } from '@angular/core';
2.  @Component({
3.    selector: 'app-hello',
4.    templateUrl: "./hello.component.html",
5.    styleUrls: ['./hello.component.css']
6.  })
7.  export class HelloComponent implements OnInit {
8.    courseName = "Angular";
9.    constructor() { }
10.   ngOnInit() {
11.   }
12.   changeName() {
13.     this.courseName = "TypeScript";
14.   }
15. }

16.
```

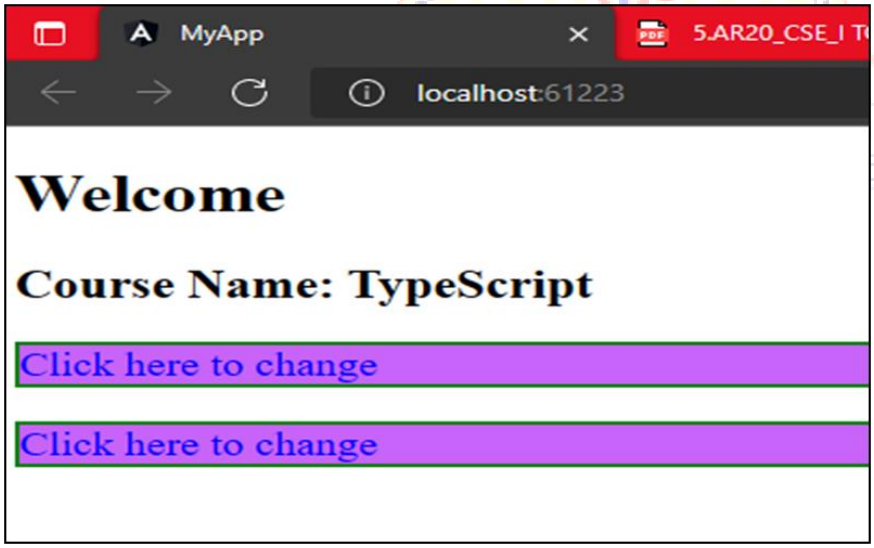
2. Open **hello.component.html** and add a paragraph and bind it with changeName() method as shown in Line 3

```
1.  <h1>Welcome</h1>
2.  <h2>Course Name: {{ courseName }}</h2>
3.  <p (click)="changeName()">Click here to change</p>

4.
```

3. Save the files and check the output in the browser

Output:


MyApp
5.AR20_CSE_I TO VIII_05
localhost:61223
Welcome
Course Name: Angular
Click here to change
Click here to change


MyApp
5.AR20_CSE_I T
localhost:61223
Welcome
Course Name: TypeScript
Click here to change
Click here to change

2.a Course Name: Angular JS
Module Name: Structural Directives - ngIf
Create a login form with username and password fields. If the user enters the correct credentials, it should render a "Welcome <<username>>" message otherwise it should render "Invalid Login!!! Please try again..." message.
Program:

Open **app.component.ts and** write the following code:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  isAuthenticated!: boolean;
  submitted = false;
  userName!: string;

  onSubmit(name: string, password: string) {
    this.submitted = true;
    this.userName = name;
    if (name === 'admin' && password === 'admin') {
      this.isAuthenticated = true;
    } else {
      this.isAuthenticated = false;
    }
  }
}
```

2. Write the below-given code in **app.component.html:**

```
<div *ngIf="!submitted">
  <form>
    <label>User Name</label>
    <input type="text" #username /><br /><br />
    <label for="password">Password</label>
    <input type="password" name="password" #password /><br />
  </form>
  <button (click)="onSubmit(username.value, password.value)">Login</button>
```

```
  </div>

<div *ngIf="submitted">
  <div *ngIf="isAuthenticated; else failureMsg">
    <h4>Welcome {{ userName }}</h4>
  </div>
  <ng-template #failureMsg>
    <h4>Invalid Login !!! Please try again...</h4>
  </ng-template>
  <button type="button" (click)="submitted = false">Back</button>

</div>
```

3. Add AppComponent to the bootstrap property in the root module file i.e., **app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

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

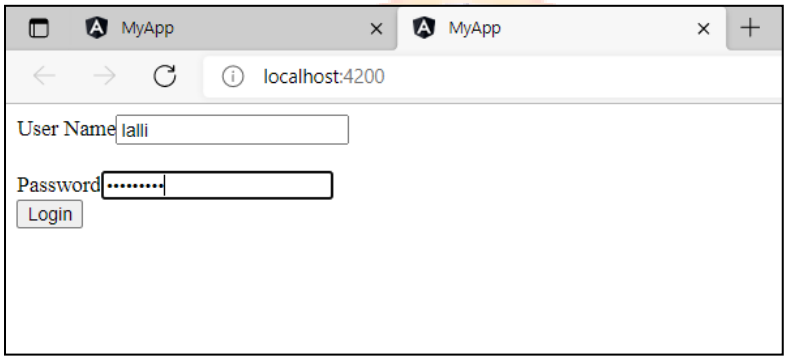
4. Ensure the index.html displays app-root.

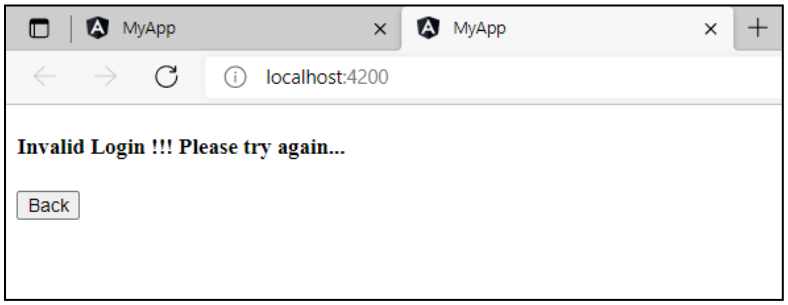
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MyApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
```

```
<body>
  <app-root></app-root>
</body>
</html>
```

5. Save the files and check the output in the **browser.**

Output:





2.b Course Name: Angular JS
Module Name: ngFor
Create a courses array and rendering it in the template using ngFor directive in a list format.
Program:1. Write the below-given code in **app.component.ts**

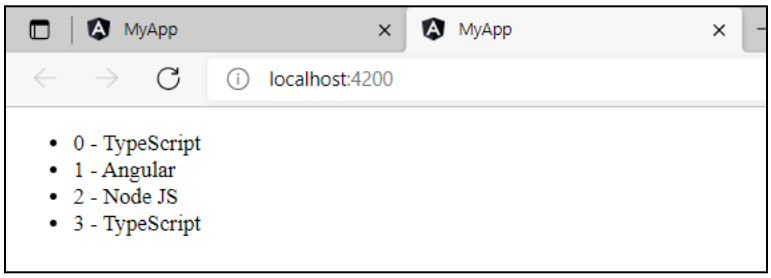
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  courses: any[] = [
    { id: 1, name: 'TypeScript' },
    { id: 2, name: 'Angular' },
    { id: 3, name: 'Node JS' },
    { id: 1, name: 'TypeScript' }
  ];
}
```

2. Write the below-given code in **app.component.html**

```
<ul>
  <li *ngFor="let course of courses; let i = index">
    {{ i }} - {{ course.name }}
  </li>
</ul>
```

3. Save the files and check the output in the browser

Output:



2.c Course Name: Angular JS
Module Name: ngSwitch
Display the correct option based on the value passed to ngSwitch directive.
Program:

1. Write the below-given code in **app.component.ts**

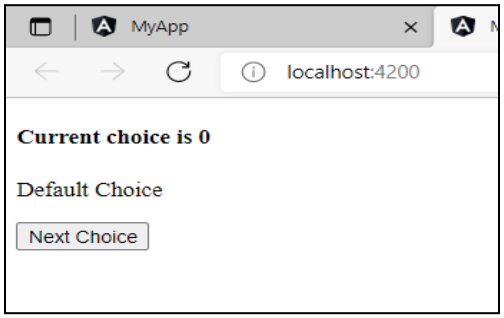
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  choice = 0;

  nextChoice() {
    this.choice++;
  }

}
```

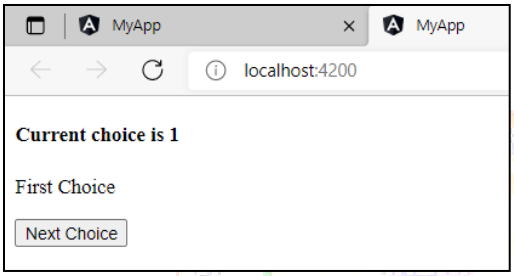
2. Write the below-given code in **app.component.html**

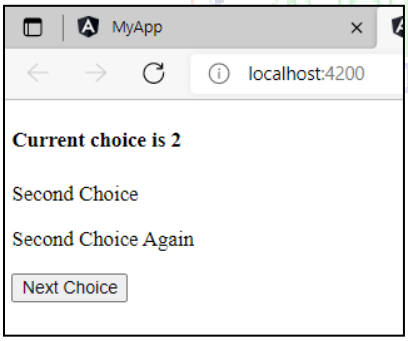
```
<h4>
  Current choice is {{ choice }}
</h4>

<div [ngSwitch]="choice">
  <p *ngSwitchCase="1">First Choice</p>
  <p *ngSwitchCase="2">Second Choice</p>
  <p *ngSwitchCase="3">Third Choice</p>
  <p *ngSwitchCase="2">Second Choice Again</p>
  <p *ngSwitchDefault>Default Choice</p>
</div>

<div>
  <button (click)="nextChoice()">
            Next Choice
      </button>
</div>
```

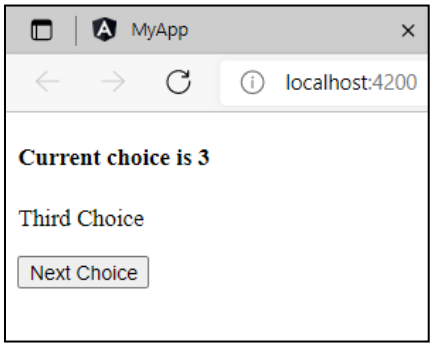
3.Save the files and check the output in the browser

Output:









2.d Course Name: Angular JS
Module Name: Custom Structural Directive
Create a custom structural directive called 'repeat' which should repeat the element given a number of times.
program:

1. Generate a directive called 'repeat' using the following command:

```
1. D:\MyApp> ng generate directive repeat
```

2. Write the below-given code in **app.module.ts**

```
1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3.
4. import { AppComponent } from './app.component';
5. import { RepeatDirective } from './repeat.directive';
6.
7. @NgModule({
8.   declarations: [
9.     AppComponent,
10.    RepeatDirective
11.  ],
12.  imports: [
13.    BrowserModule
14.  ],
15.  providers: [],
16.  bootstrap: [AppComponent]
17. })

18. export class AppModule { }
```

3. Open the **repeat.directive.ts** file and add the following code

```
1. import { Directive, TemplateRef, ViewContainerRef, Input } from '@angular/core';
2.
3. @Directive({
4.   selector: '[appRepeat]'
```

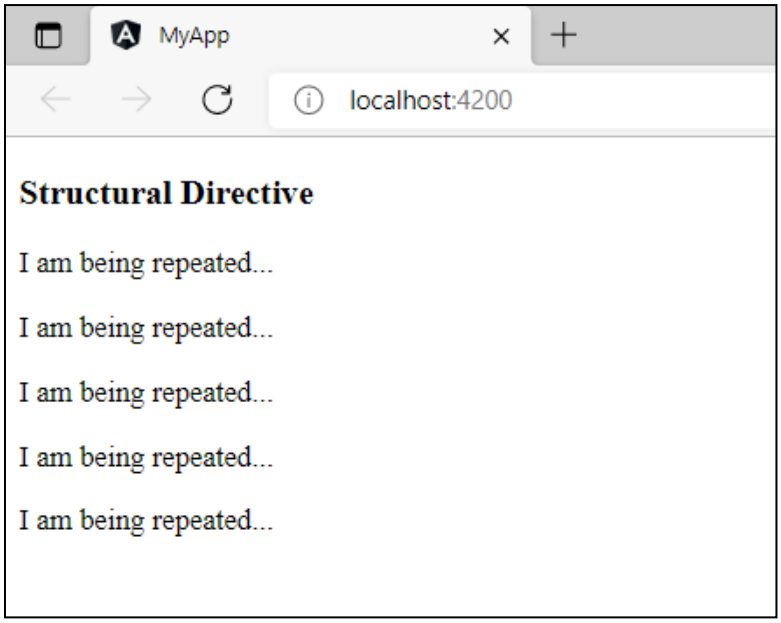
```
})
export class RepeatDirective {

  constructor(private templateRef: TemplateRef<any>, private viewContainer: ViewContainerRef) { }

  @Input() set appRepeat(count: number) {
    for (let i = 0; i < count; i++) {
      this.viewContainer.createEmbeddedView(this.templateRef);
    }
  }
}
```

4. Write the below-given code in **app.component.html**

```
<h3>Structural Directive</h3>

<p *appRepeat="5">I am being repeated...</p>
```

6. Save the files and check the output in the browser

Output:



3.a Course Name: Angular JS
Module Name: Attribute Directives - ngStyle
Apply multiple CSS properties to a paragraph in a component using ngStyle.
Program:

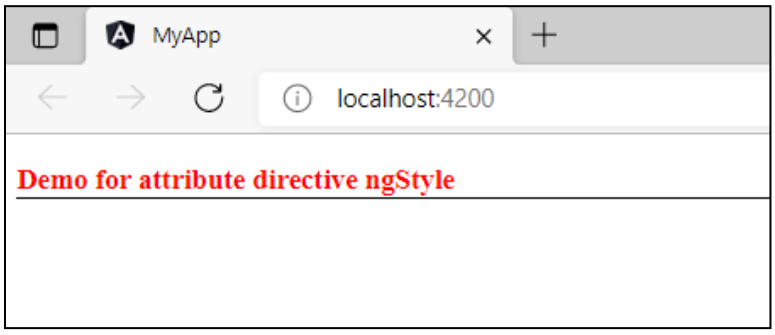
1. Write the below-given code in **app.component.ts**

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  colorName = 'red';
  fontWeight = 'bold';
  borderStyle = '1px solid black';
}
```

2. Write the below-given code in **app.component.html**

```
<p [ngStyle]="{
            color:colorName,
            'font-weight':fontWeight,
            borderBottom: borderStyle
        }">
  Demo for attribute directive ngStyle
</p>
```

Output:



3.b Course Name: Angular JS
Module Name: ngClass
Apply multiple CSS classes to the text using ngClass directive.
Program:1. Write the below-given code in **app.component.ts**

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

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

2. Write the below-given code in **app.component.html**

```
<div [ngClass]="{bordered: isBordered}">
  Border {{ isBordered ? "ON" : "OFF" }}
</div>
```

3. In **app.component.css**, add the following CSS class

```
.bordered {
        border: 1px dashed black;
        background-color: #eee;
}
```

output:

3.c Course Name: Angular JS
Module Name: Custom Attribute Directive
Create an attribute directive called 'show Message' which should display the given message in a paragraph when a user clicks on it and should change the text color to red.
Program:

1. Generate a directive called 'message' using the following command

```
1. D:\MyApp>ng generate directive message
```

2. Above command will add MessageDirective class to the declarations property in the **app.module.ts** file

```
1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3.
4. import { AppComponent } from './app.component';
5. import { MessageDirective } from './message.directive';
6.
7. @NgModule({
8.   declarations: [
9.     AppComponent,
10.    MessageDirective
11.  ],
12.  imports: [
13.    BrowserModule
14.  ],
15.  providers: [],
16.  bootstrap: [AppComponent]
17. })
18. export class AppModule { }
```

3. Open the **message.directive.ts** file and add the following code

```
1. import { Directive, ElementRef, Renderer2, HostListener, Input } from '@angular/core';
2.
3. @Directive({
4.   selector: '[appMessage]',
5. })
6. export class MessageDirective {
7.   @Input('appMessage') message!: string;
```

```

  constructor(private el: ElementRef, private renderer: Renderer2) {
    renderer.setStyle(el.nativeElement, 'cursor', 'pointer');
  }

  @HostListener('click') onClick() {
    this.el.nativeElement.innerHTML = this.message;
    this.renderer.setStyle(this.el.nativeElement, 'color', 'red');

  }}
```

4. Write the below-given code in **app.component.html**

```
<h3>Attribute Directive</h3>

<p [appMessage]="myMessage">Click Here</p>
```

5. Add the following CSS styles to the **app.component.css** file

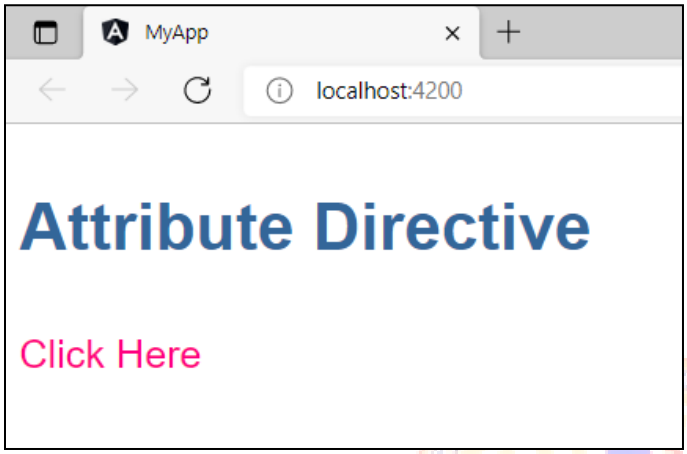
```
h3 {
  color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
}
p {
  color: #ff0080;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 150%;

}
```

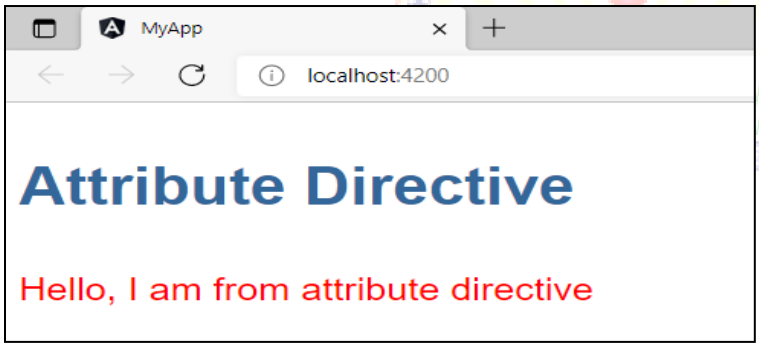
6. Add the following code in **app.component.ts**

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

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

```
    myMessage = 'Hello, I am from attribute directive';
}

```

Output:





4.a Course Name: Angular JS
Module Name: Property Binding
Binding image with class property using property binding.
Program:

Write the following code in **app.component.ts** as shown below

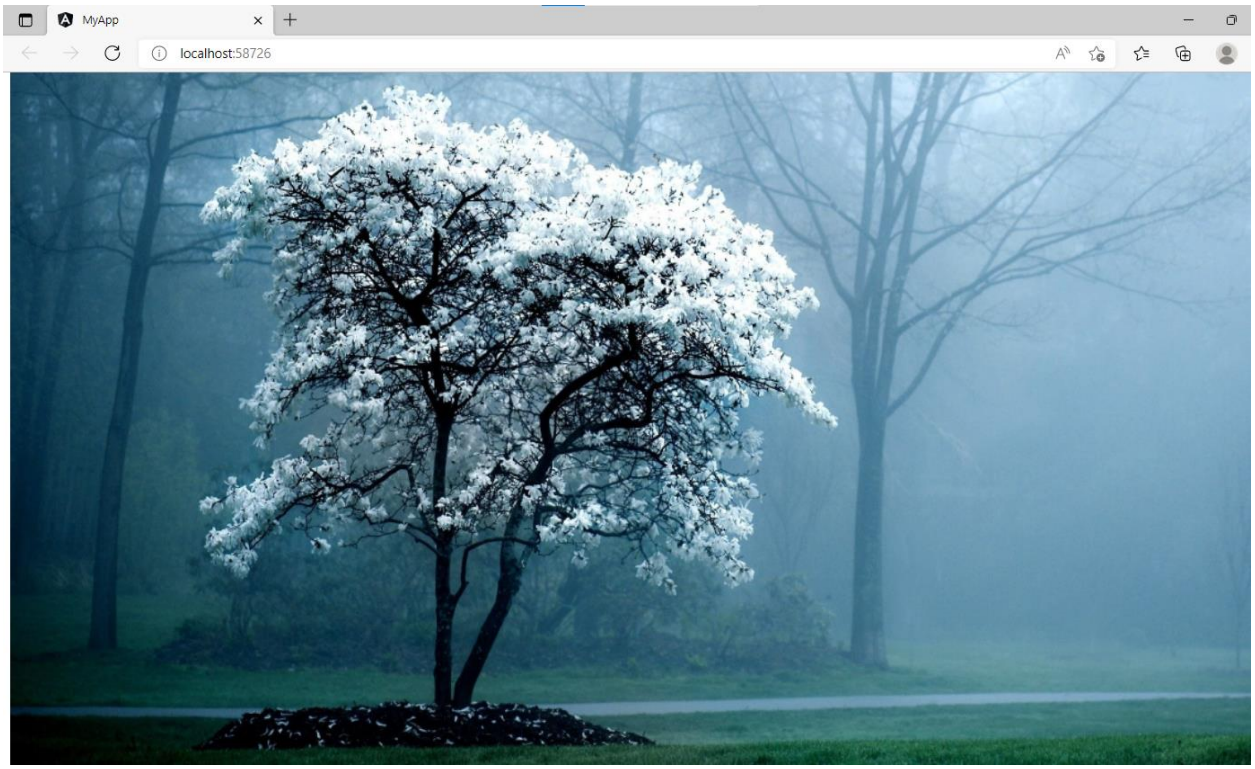
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  imgUrl = 'assets/imgs/logo.png';
}
```

Create a folder named "imgs" inside src/assets and place a logo.png file inside it.

2. Write the following code in **app.component.html** as shown below

```
<img [src]='imgUrl'>
```

2. Save the files and check the output in the browser.


MyApp
localhost:58726

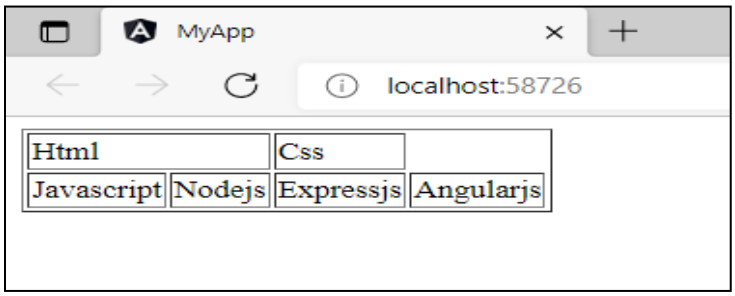
4.b Course Name: Angular JS
Module Name: Attribute Binding
Binding colspan attribute of a table element to the class property.
Program:1. Write the below-given code in **app.component.ts**

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  colspanValue = '2';
}
```

2. Write the below-given code in **app.component.html**

```
<table border=1>
    <tr>
        <td [attr.colspan]="colspanValue"> Html </td>
        <td>Css</td>
    </tr>
    <tr>
        <td>Javascript</td>
        <td>nodejs</td>
        <td>Expressjs</td>
      <td>Angularjs</td>

    </tr>
</table>
```

3. Save the files and check the output in the browser
4. **Output:**



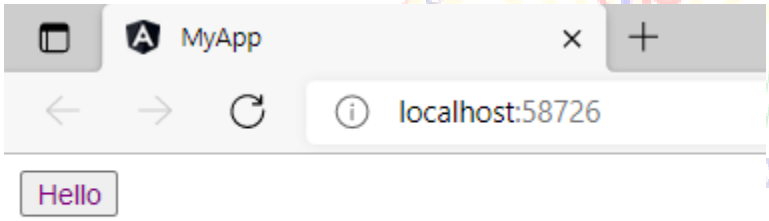
4.c Course Name: Angular JS
Module Name: Style and Event Binding
Binding an element using inline style and user actions like entering text in input fields.
program:

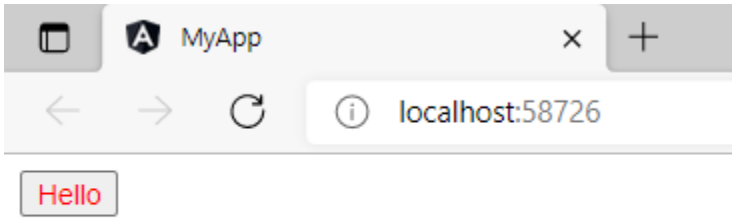
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  colspanValue = '2';
  isValid="purple";
}
```

App.component .html:

```
<button [style.color] = "isValid ? 'purple' : 'red' ">Hello</button>
```

Output:

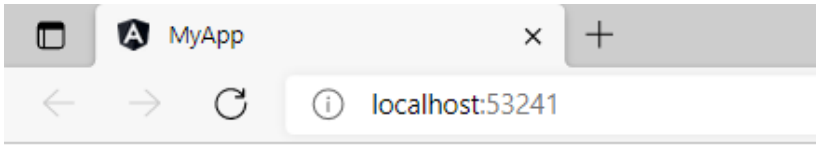




5.a Course Name: Angular JS
Module Name: Built in Pipes
Display the product code in lowercase and product name in uppercase using built-in pipes.
PROGRAM:

```
<h3> {{ title | titlecase}} </h3>
<table style="text-align:left">
   <tr>
      <th> Product Code1</th>
      <td> {{ productCode1| lowercase }} </td>
   </tr>
   <tr>
      <th> Product Name1 </th>
      <td> {{ productName1 | uppercase }} </td>
   </tr>
   <tr>
      <th> Product Code2 </th>
      <td> {{ productCode2| lowercase }} </td>
   </tr>
   <tr>
      <th> Product Name2 </th>
      <td> {{ productName2 | uppercase }} </td>
   </tr>
</table>

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'product details';
  productCode1 = 'PROD_P001';
  productName1= 'Laptop';
  productCode2= 'PROD_P002';
  productName2= 'Computer';
}
```

OUTPUT:



Product Details

Product Code1 prod_p001
Product Name1 LAPTOP
Product Code2 prod_p002
Product Name2 COMPUTER

5.b Course Name: Angular JS
Module Name: Passing Parameters to Pipes
Apply built-in pipes with parameters to display product details.
Program:

1. Write the below-given code in app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'product details';
  productCode = 'PROD_P001';
  productName = 'Apple MPTT2 MacBook Pro';
  productPrice = 217021;
  purchaseDate = '1/17/2018';
  productTax = '0.1';
  productRating = 4.92;
}
```

2. Write the below-given code in app.component.html

```
<h3> {{ title | titlecase}} </h3>
<table style="text-align:left">
    <tr>
        <th> Product Code </th>
        <td> {{ productCode | slice:5:9 }} </td>
    </tr>
    <tr>
        <th> Product Name </th>
        <td> {{ productName | uppercase }} </td>
    </tr>
    <tr>
        <th> Product Price </th>
        <td> {{ productPrice | currency: 'INR':'symbol':":'fr' }} </td>
    </tr>
    <tr>
        <th> Purchase Date </th>
        <td> {{ purchaseDate | date:'fullDate' | lowercase}} </td>
    </tr>
    <tr>
        <th> Product Tax </th>
        <td> {{ productTax | percent : '.2' }} </td>
    </tr>
    <tr>
        <th> Product Rating </th>
```

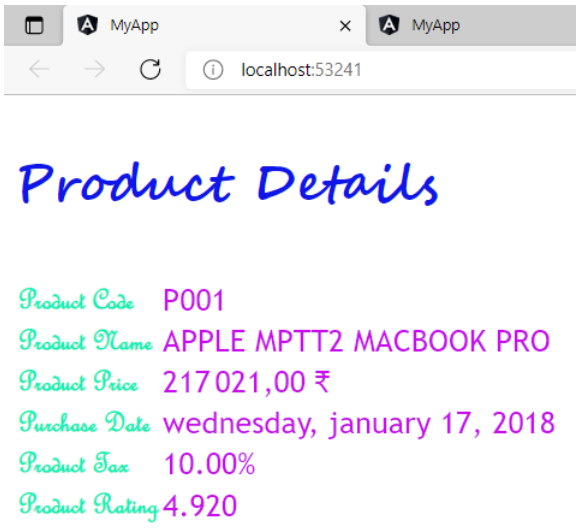
```
      <td>{{ productRating | number:'1.3-5'}} </td>
    </tr>
</table>
```

3. Write the below-given code in app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { registerLocaleData } from '@angular/common';
import localeFrench from '@angular/common/locales/fr';
registerLocaleData(localeFrench);
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

4. Save the files and check the output in the browser.

Output:



5.c Course Name: Angular JS
Module Name: Nested Components Basics
Load CourseslistComponent in the root component when a user clicks on the View courses list button.
Program:

1. Create a component called coursesList using the following CLI command

```
D:\MyApp>ng generate component coursesList
```

The above command will create a folder with name courses-list with the following files

- courses-list.component.ts
- courses-list.component.html
- courses-list.component.css
- courses-list.component.spec.ts

2. CoursesListComponent class will be added in the **app.module.ts** file

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { CoursesListComponent } from './courses-list/courses-list.component';

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

```

3. Write the below-given code in **courses-list.component.ts**

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-courses-list',
  templateUrl: './courses-list.component.html',
  styleUrls: ['./courses-list.component.css']
})
export class CoursesListComponent {
  courses = [
    { courseId: 1, courseName: "Node JS" },
    { courseId: 2, courseName: "Typescript" },
    { courseId: 3, courseName: "Angular" },
    { courseId: 4, courseName: "React JS" }

  ];}
```

4. Write the below-given code in **courses-list.component.html**

```
<table border="1">
  <thead>
    <tr>
      <th>Course ID</th>
      <th>Course Name</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let course of courses">
      <td>{{ course.courseId }}</td>
      <td>{{ course.courseName }}</td>
    </tr>
  </tbody>
</table>
```

5. Add the following code in **courses-list.component.css**

```
tr{
    text-align:center;
}
```

6. Write the below-given code in **app.component.html**

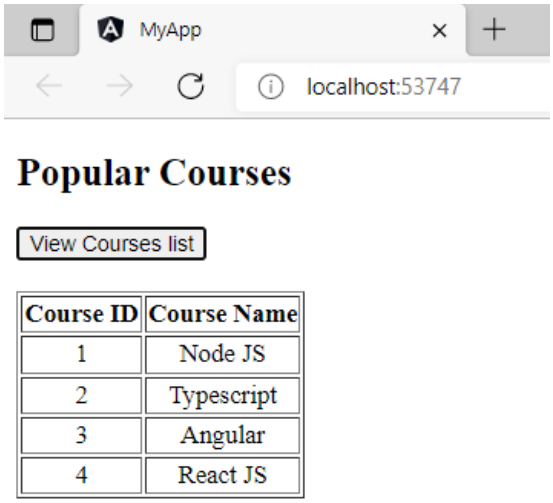
```
<h2>Popular Courses</h2>
<button (click)="show = true">View Courses list</button><br /><br />
<div *ngIf="show">
  <app-courses-list></app-courses-list>
</div>
```

7. Write the below-given code in **app.component.ts**

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

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

}
```

8. Save the files and check the output in the browser

Output:



6.a Course Name: Angular JS Module Name: Passing data from Container Component to Child Component Create an AppComponent that displays a dropdown with a list of courses as values in it. Create another component called the CoursesList component and load it in AppComponent which should display the course details. When the user selects a course from the dropdown, corresponding course details should be loaded.
Program:

1. Open the **courses-list.component.ts** file created in the example of nested components and add the following code

```
import { Component, Input } from '@angular/core';
@Component({
  selector: 'app-courses-list',
  templateUrl: './courses-list.component.html',
  styleUrls: ['./courses-list.component.css'],
})
export class CoursesListComponent {
  courses = [
    { courseId: 1, courseName: 'Node JS' },
    { courseId: 2, courseName: 'Typescript' },
    { courseId: 3, courseName: 'Angular' },
    { courseId: 4, courseName: 'React JS' },
  ];
  course!: any[];
  @Input() set cName(name: string) {
    this.course = [];
    for (var i = 0; i < this.courses.length; i++) {
      if (this.courses[i].courseName === name) {
        this.course.push(this.courses[i]);
      }
    }
  }
}
```

2. Open **courses-list.component.html** and add the following code

```
<table border="1" *ngIf="course.length > 0">
  <thead>
    <tr>
      <th>Course ID</th>
      <th>Course Name</th>
    </tr>
  </thead>
```

```
  <tbody>
    <tr *ngFor="let c of course">
      <td>{{ c.courseId }}</td>
      <td>{{ c.courseName }}</td>
    </tr>
  </tbody>
</table>
```

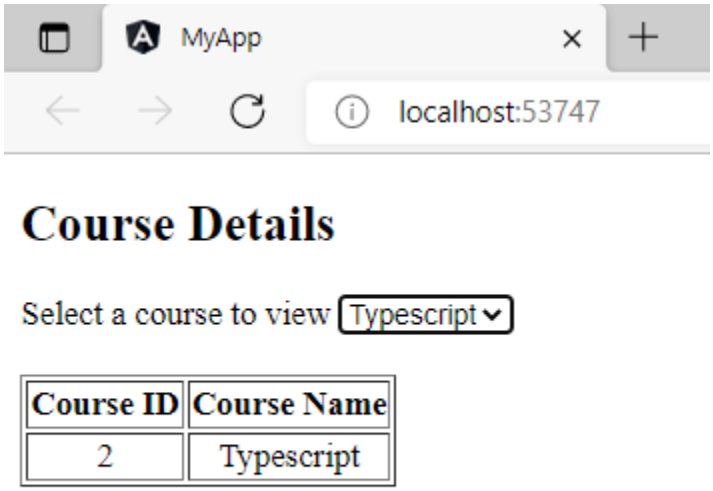
3. Add the following in **app.component.html**

```
<h2>Course Details</h2>

Select a course to view
<select #course (change)="name = course.value">
  <option value="Node JS">Node JS</option>
  <option value="Typescript">Typescript</option>
  <option value="Angular">Angular</option>
  <option value="React JS">React JS</option></select><br /><br />

<app-courses-list [cName]="name"></app-courses-list>
```

4. Add the following in **app.component.ts**

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

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

Output:


MyApp
localhost:53747
Course Details
Select a course to view
Typescript
Course ID
Course Name
2
Typescript

6.b Course Name: Angular JS Module Name: Passing data from Child Component to ContainerComponent Create an AppComponent that loads another component called the CoursesList component. Create another component called CoursesListComponent which should display the courses list in a table along with a register .button in each row. When a user clicks on the register button, it should send that courseName value back to AppComponent where it should display the registration successful message along with courseName.

Program:

1. Open the **courses-list.component.ts** file created in the previous example and add the following code

```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-courses-list',
  templateUrl: './courses-list.component.html',
  styleUrls: ['./courses-list.component.css']
})
export class CoursesListComponent {
  @Output() registerEvent = new EventEmitter<string>();
  courses = [
    { courseId: 1, courseName: 'Node JS' },
    { courseId: 2, courseName: 'Typescript' },
    { courseId: 3, courseName: 'Angular' },
    { courseId: 4, courseName: 'React JS' }
  ];
  register(courseName: string) {
    this.registerEvent.emit(courseName);
  }
}
```

2. Open **courses-list.component.html** and add the following code

```
<table border="1">
  <thead>
    <tr>
      <th>Course ID</th>
      <th>Course Name</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let course of courses">
      <td>{{ course.courseId }}</td>
```

```
      <td>{{ course.courseName }}</td>
      <td><button (click)="register(course.courseName)">Register</button></td>
    </tr>
  </tbody>
</table>
```

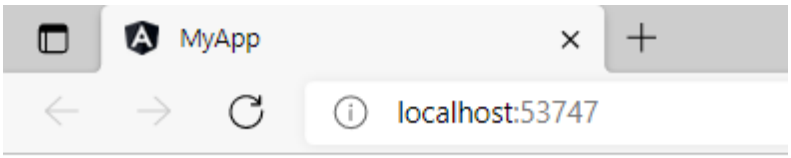
3. Add the following in **app.component.html**

```
<h2>Courses List</h2>

<app-courses-list (registerEvent)="courseReg($event)"></app-courses-list>
<br /><br />

<div *ngIf="message">{{ message }}</div>
```

4. Add the following code in **app.component.ts**

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  message!: string;
  courseReg(courseName: string) {
    this.message = `Your registration for ${courseName} is successful`;
  }
}
```

Output:



Courses List

Course ID	Course Name	
1	Node JS	Register
2	Typescript	Register
3	Angular	Register
4	React JS	Register

Your registration for Angular is successful

6.c Course Name: Angular JS
Module Name: Shadow DOM
Apply ShadowDOM and None encapsulation modes to components.
Program:

Create a component called **First** using the following CLI command

```
1. D:\MyApp>ng generate component first
```

first.component.css

```
1. .cmp {
2.     padding: 6px;
3.     margin: 6px;
4.     border: blue 2px solid;
5.   }
```

first.component.html

```
1. <div class="cmp">First Component</div>
```

Line 1: CSS class called cmp is applied to the div tag

Create a component called **Second** using the following CLI command

```
D:\MyApp>ng generate component second
```

second.component.css

```
1. .cmp {
2.   border: green 2px solid;
3.   padding: 6px;
4.   margin: 6px;
5. }
```

second.component.html

```
1. <div class="cmp">Second Component</div>
```

Line 1: CSS class called cmp is applied to the div tag

app.component.css

```
1. .cmp {
2.     padding: 8px;
3.     margin: 6px;
4.     border: 2px solid red;
5.   }
```

app.component.html

```
1. <h3>CSS Encapsulation with Angular</h3>
2. <div class="cmp">
3.     App Component
4.     <app-first></app-first>
5.     <app-second></app-second>
6. </div>
```

Line 2: Apply CSS class called cmp

Line 4: Load first component

Line 5: load second component

app.component.ts

```
1. import { Component } from '@angular/core';
2.
3. @Component({
4.   selector: 'app-root',
5.   templateUrl: './app.component.html',
6.   styleUrls: ['./app.component.css']
7. })
8. export class AppComponent {
```

```
9.
10. }

11.
```

first.component.ts

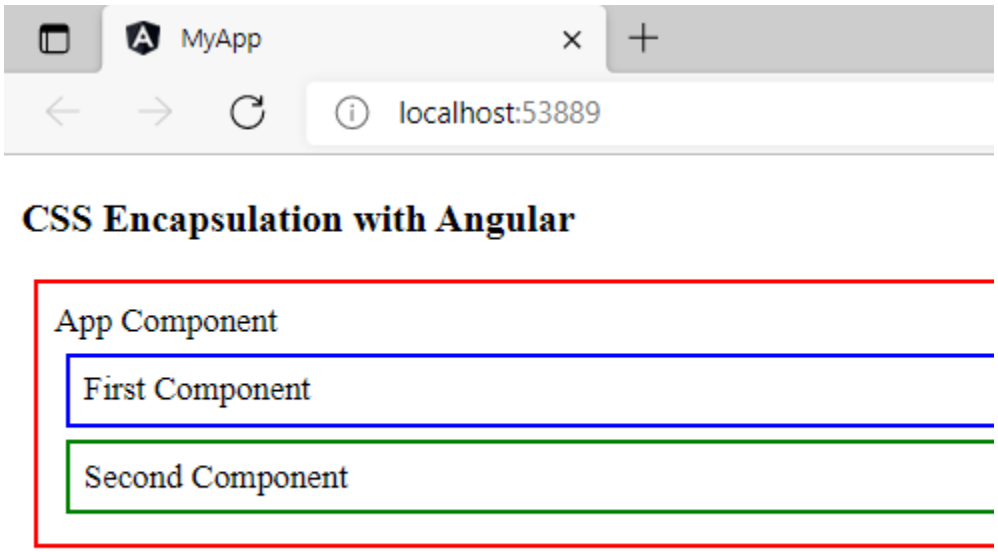
```
1.  import { Component } from '@angular/core';
2.
3.  @Component({
4.    selector: 'app-first',
5.    templateUrl: './first.component.html',
6.    styleUrls: ['./first.component.css']
7.  })
8.  export class FirstComponent {
9.
10. }

11.
```

second.component.ts

```
1.  import { Component, ViewEncapsulation } from '@angular/core';
2.  @Component({
3.    selector: 'app-second',
4.    templateUrl: './second.component.html',
5.    styleUrls: ['./second.component.css'],
6.    encapsulation: ViewEncapsulation.ShadowDom
7.  })
8.  export class SecondComponent {
9.  }
```

Line 7: encapsulation property sets the encapsulation mode of SecondComponent to ShadowDom

Output:



Program:

1. Set ViewEncapsulation to none mode in **app.component.ts** file

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

@Component({
  selector: 'app-root',
  styleUrls: ['./app.component.css'],
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
}
```

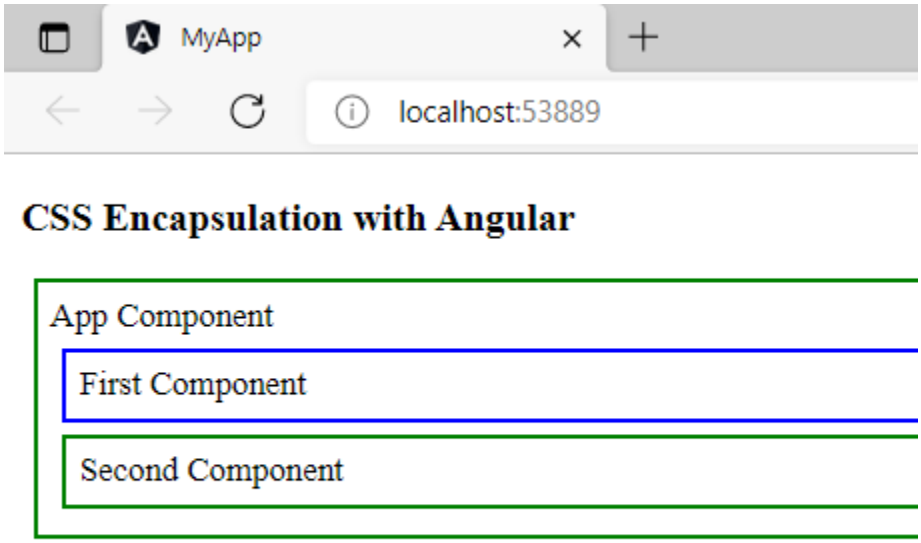
2. Set ViewEncapsulation to none mode in **second.component.ts** file

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

@Component({
  selector: 'app-second',
  templateUrl: './second.component.html',
  styleUrls: ['./second.component.css'],
```

```
    encapsulation: ViewEncapsulation.None
})
export class SecondComponent {

}
```

3. Save the files and check the output in the browser

Output:



6.d Course Name: Angular JS
Module Name: Component Life Cycle
Override component life-cycle hooks and logging the corresponding messages to understand the flow.
Program:

. Write the below-given code in **app.component.ts**

```
import {
    Component, OnInit, DoCheck, AfterContentInit, AfterContentChecked,
    AfterViewInit, AfterViewChecked,
    OnDestroy
} from '@angular/core';
@Component({
    selector: 'app-root',
    styleUrls: ['./app.component.css'],
    templateUrl: './app.component.html'
})
export class AppComponent implements OnInit, DoCheck,
    AfterContentInit, AfterContentChecked,
    AfterViewInit, AfterViewChecked,
    OnDestroy {
    data = 'Angular';
    ngOnInit() {
        console.log('Init');
    }
    ngDoCheck(): void {
        console.log('Change detected');
    }
    ngAfterContentInit(): void {
        console.log('After content init');
    }
    ngAfterContentChecked(): void {
        console.log('After content checked');
    }
    ngAfterViewInit(): void {
        console.log('After view init');
    }
    ngAfterViewChecked(): void {
        console.log('After view checked');
    }
    ngOnDestroy(): void {
        console.log('Destroy');
    }
}
```

2. Write the below-given code in **app.component.html**

```
<div>
  <h1>I'm a container component</h1>
  <input type="text" [(ngModel)]="data" />
  <app-child [title]="data"></app-child>
</div>

```

3. Write the below-given code in **child.component.ts**

```
import { Component, OnChanges, Input } from '@angular/core';
@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css']
})
export class ChildComponent implements OnChanges {
  @Input() title!: string;
  ngOnChanges(changes: any): void {
    console.log('changes in child:' + JSON.stringify(changes));
  }

}
```

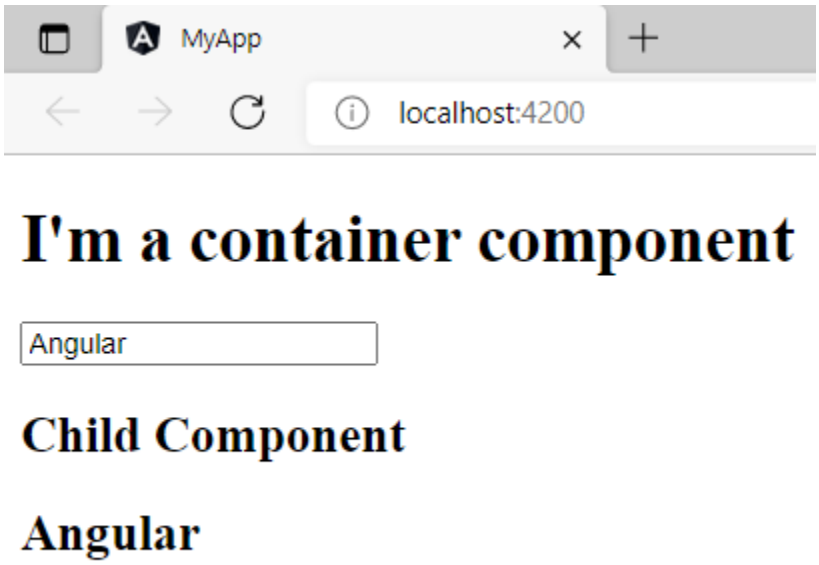
4. Write the below-given code in **child.component.html**

```
<h2>Child Component</h2>

<h2>{{title}}</h2>
```

5. Ensure FormsModule is present in the imports section of the AppModule.

6. Save the files and check the output in the browser

Output:


MyApp
localhost:4200
I'm a container component
Angular
Child Component
Angular

7.a Course Name: Angular JS
Module Name: Template Driven Forms .
Create a course registration form as a template-driven form.
Program:

App.component.html:

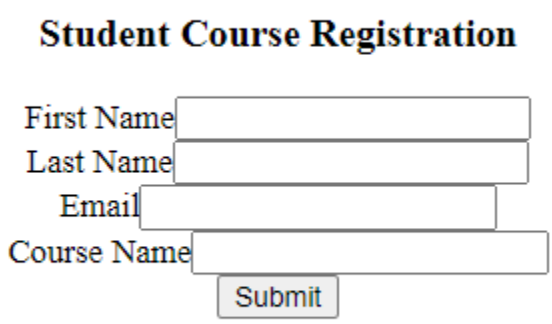
```
<br/>
<div class="container">
  <div class="row">
     <div class="form-bg">
        <form #studentForm="ngForm" (ngSubmit)="RegisterStudent(studentForm)"
align="center">
          <div class="panel panel-primary">
            <div class="panel-heading">
              <h3 class="panel-title"> Student Course Registration</h3>
            </div>
            <div class="panel-body">

              <div class="form-group">
                <label for="firstName">First Name</label>
                <input id="firstName" type="text" class="form-control"
                     name="firstName" ngModel>
              </div>
              <div class="form-group">
                <label for="lastName">Last Name</label>
                <input id="lastName" type="text" class="form-control"
                     name="lastName" ngModel>
              </div>
              <div class="form-group">
                <label for="email">Email</label>
                <input id="email" type="text" class="form-control"
                     name="email" ngModel>
              </div>
              <div class="form-group">
                <label for="course">Course Name</label>
                <input id="course" type="text" class="form-control"
                     name="Course Name" ngModel>
              </div>
            </div>
            <div class="panel-footer">
              <button class="btn btn-primary" type="submit">Submit</button>
            </div>
          </div>
        </form>
     </div>
  </div>
```

```
</div>
```

App.component.ts:

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms'
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  RegisterStudent(studentForm: NgForm): void {
    console.log(studentForm.value);
  }
}
```

Output:



7.b Course Name: Angular JS
Module Name: Model Driven Forms or Reactive Forms
Create an employee registration form as a reactive form.
Program:

1.Write the below-given code in **app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { RegistrationFormComponent } from './registration-form/registration-form.component';

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

```

2. Create a component called **RegistrationForm** using the following CLI command

```
ng generate component RegistrationForm
```

3. Add the following code in the **registration-form.component.ts** file

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
```

```
  selector: 'app-registration-form',
  templateUrl: './registration-form.component.html',
  styleUrls: ['./registration-form.component.css']
})
export class RegistrationFormComponent implements OnInit {

  registerForm!: FormGroup;
  submitted!:boolean;

  constructor(private formBuilder: FormBuilder) { }

  ngOnInit() {
    this.registerForm = this.formBuilder.group({
      firstName: [", Validators.required],
      lastName: [", Validators.required],
      address: this.formBuilder.group({
        street: [],
        zip: [],
        city: []
      })
    });
  }

}

```

4. Write the below-given code in **registration-form.component.html**

```
<div class="container">
    <h1>Registration Form</h1>
    <form [formGroup]="registerForm">
      <div class="form-group">
        <label>First Name</label>
        <input type="text" class="form-control" formControlName="firstName">
            <div *ngIf="registerForm.controls['firstName'].errors" class="alert alert-danger">
            Firstname field is invalid.
            <p *ngIf="registerForm.controls['firstName'].errors?.['required']">
                This field is required!
            </p>
        </div>
      </div>
      <div class="form-group">
        <label>Last Name</label>
        <input type="text" class="form-control" formControlName="lastName">
```

```
      <div *ngIf="registerForm.controls['lastName'].errors" class="alert alert-danger">
          Lastname field is invalid.
          <p *ngIf="registerForm.controls['lastName'].errors?.['required']">
              This field is required!
          </p>
      </div>
    </div>
    <div class="form-group">
      <fieldset formGroupName="address">
        <legend>Address:</legend>
        <label>Street</label>
        <input type="text" class="form-control" formControlName="street">
        <label>Zip</label>
        <input type="text" class="form-control" formControlName="zip">
        <label>City</label>
        <input type="text" class="form-control" formControlName="city">
      </fieldset>
    </div>
    <button type="submit" class="btn btn-primary" (click)="submitted=true">Submit</button>
  </form>
 <br/>
  <div [hidden]="!submitted">
    <h3> Employee Details </h3>
    <p>First Name: {{ registerForm.get('firstName')?.value }} </p>
    <p> Last Name: {{ registerForm.get('lastName')?.value }} </p>
    <p> Street: {{ registerForm.get('address.street')?.value }}</p>
    <p> Zip: {{ registerForm.get('address.zip')?.value }} </p>
    <p> City: {{ registerForm.get('address.city')?.value }}</p>
  </div>
 </div>

```

5. Write the below-given code in **registration-form.component.css**

```
.ng-valid[required]  {
  border-left: 5px solid #42A948; /* green */
}

.ng-invalid:not(form)  {
  border-left: 5px solid #a94442; /* red */
}
```

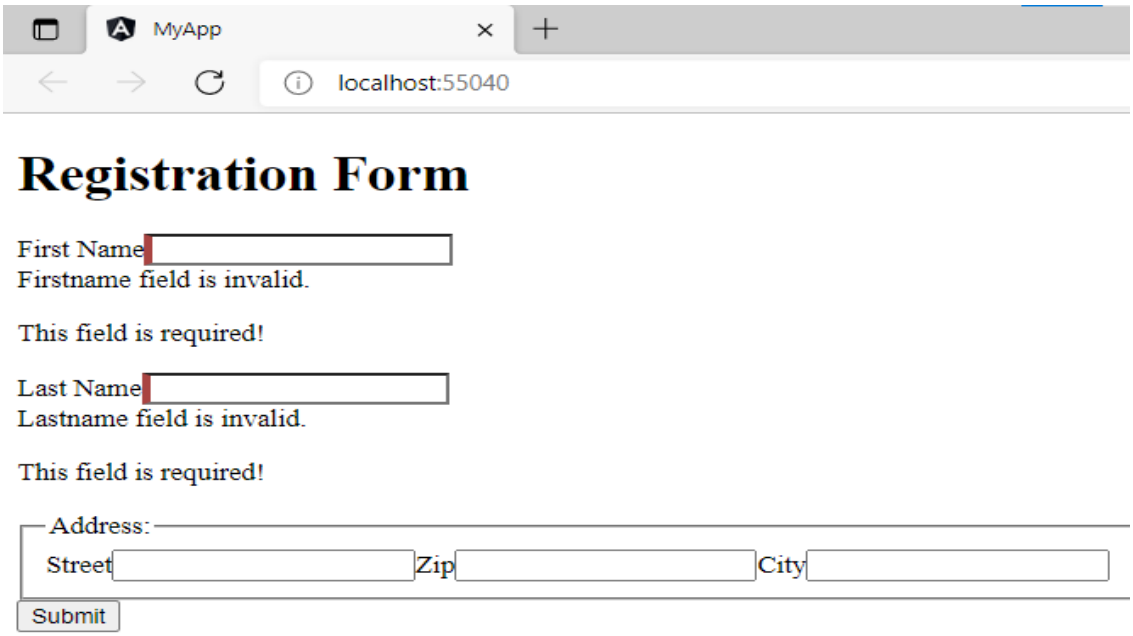
```
8.
```

6. Write the below-given code in **app.component.html**

```
1. <app-registration-form></app-registration-form>
```

9. Save the files and check the output in the browser.

Output:



7.c Course Name: Angular JS
Module Name: Custom Validators in Reactive Forms
Create a custom validator for an email field in the employee registration form .(reactive form).
program:

1. Write a separate function in **registration-form.component.ts** for custom validation as shown below.

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormControl, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-registration-form',
  templateUrl: './registration-form.component.html',
  styleUrls: ['./registration-form.component.css']
})
export class RegistrationFormComponent implements OnInit {

  registerForm!: FormGroup;
  submitted!:boolean;

  constructor(private formBuilder: FormBuilder) { }

  ngOnInit() {
    this.registerForm = this.formBuilder.group({
      firstName: ['',Validators.required],
      lastName: ['', Validators.required],
      address: this.formBuilder.group({
        street: [],
        zip: [],
        city: []
      }),
      email: ['',[Validators.required,validateEmail]]
    });
  }

}
function validateEmail(c: FormControl): any {
  let EMAIL_REGEXP = /^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$/;

  return EMAIL_REGEXP.test(c.value) ? null : {
    emailInvalid: {
      message: "Invalid Format!"
    }
```

```
};

}
```

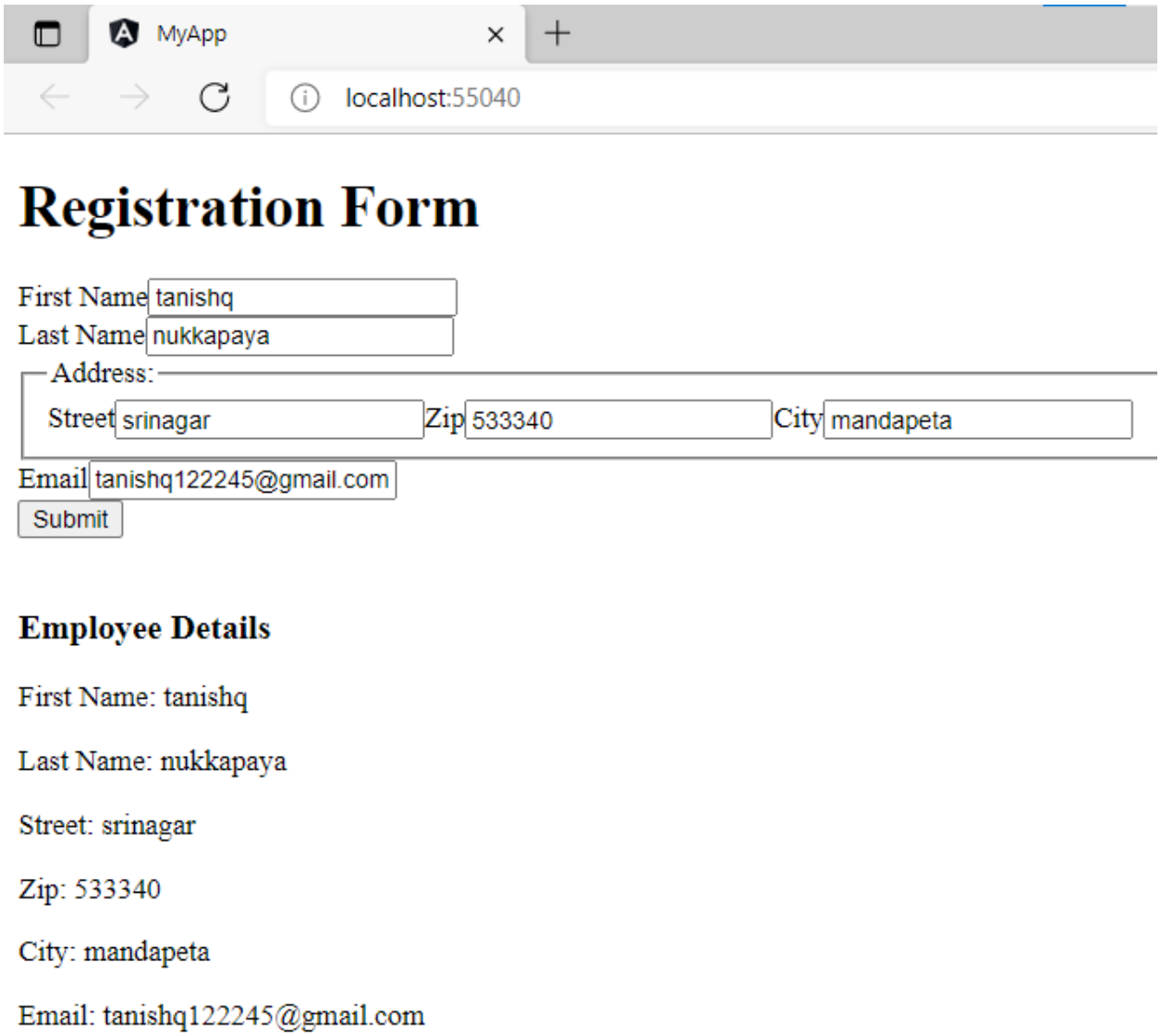
2. Add HTML controls for the email field in the **registration-form.component.html** file as shown below

```
<div class="container">
  <h1>Registration Form</h1>
  <form [formGroup]="registerForm">
    <div class="form-group">
      <label>First Name</label>
      <input type="text" class="form-control" formControlName="firstName">
        <div *ngIf="registerForm.controls['firstName'].errors" class="alert alert-danger">
        Firstname field is invalid.
        <p *ngIf="registerForm.controls['firstName'].errors?.['required']">
          This field is required!
        </p>
      </div>
    </div>
    <div class="form-group">
      <label>Last Name</label>
      <input type="text" class="form-control" formControlName="lastName">
     <div *ngIf="registerForm.controls['lastName'].errors" class="alert alert-danger">
        Lastname field is invalid.
        <p *ngIf="registerForm.controls['lastName'].errors?.['required']">
          This field is required!
        </p>
      </div>
    </div>
    <div class="form-group">
      <fieldset formGroupName="address">
        <legend>Address:</legend>
        <label>Street</label>
        <input type="text" class="form-control" formControlName="street">
        <label>Zip</label>
        <input type="text" class="form-control" formControlName="zip">
        <label>City</label>
        <input type="text" class="form-control" formControlName="city">
      </fieldset>
    </div>
    <div class="form-group">
      <label>Email</label>
      <input type="text" class="form-control" formControlName="email" />
      <div *ngIf="registerForm.controls['email'].errors" class="alert alert-danger">
```

```
        Email field is invalid.
        <p *ngIf="registerForm.controls['email'].errors?.['required']">
          This field is required!
        </p>
        <p *ngIf="registerForm.controls['email'].errors?.['emailInvalid']">
          {{ registerForm.controls['email'].errors?.['emailInvalid'].message }}
        </p>
      </div>
      </div>
      <button type="submit" class="btn btn-primary" (click)="submitted=true">Submit</button>
    </form>
  <br/>
    <div [hidden]="!submitted">
      <h3> Employee Details </h3>
      <p>First Name: {{ registerForm.get('firstName')?.value }} </p>
      <p> Last Name: {{ registerForm.get('lastName')?.value }} </p>
      <p> Street: {{ registerForm.get('address.street')?.value }}</p>
      <p> Zip: {{ registerForm.get('address.zip')?.value }} </p>
      <p> City: {{ registerForm.get('address.city')?.value }}</p>
      <p>Email: {{ registerForm.get('email')?.value }}</p>
    </div>
  </div>
```

3. Save the files and check the output in the browser

Output:


MyApp
localhost:55040
Registration Form
First Name tanishq
Last Name nukkapaya
Address:
Street srinagar
Zip 533340
City mandapeta
Email tanishq122245@gmail.com
Submit
Employee Details
First Name: tanishq
Last Name: nukkapaya
Street: srinagar
Zip: 533340
City: mandapeta
Email: tanishq122245@gmail.com

8.b Course Name: Angular JS Module Name: Services Basics Create a Book Component which fetches book details like id, name and displays them on the page in a list format. Store the book details in an array and fetch the data using a custom service.
Program:

Create **BookComponent** by using the following CLI command

```
D:\MyApp>ng generate component book
```

2. Create a file with the name **book.ts** under the book folder and add the following code.

```
export class Book {
    id!: number;
    name!: string;
}
```

3. Create a file with the name **books-data.ts** under the book folder and add the following code.

```
import { Book } from './book';

export let BOOKS: Book[] = [
    { id: 1, name: 'HTML 5' },
    { id: 2, name: 'CSS 3' },
    { id: 3, name: 'Java Script' },
    { id: 4, name: 'Ajax Programming' },
    { id: 5, name: 'jQuery' },
    { id: 6, name: 'Mastering Node.js' },
    { id: 7, name: 'Angular JS 1.x' },
    { id: 8, name: 'ng-book 2' },
    { id: 9, name: 'Backbone JS' },
    { id: 10, name: 'Yeoman' }
];
```

4. Create a service called **BookService** under the book folder using the following CLI command

```
D:\MyApp\src\app\book>ng generate service book
```

5. Add the following code in **book.service.ts**

```
import { Injectable } from '@angular/core';
import { BOOKS } from './books-data';

@Injectable({
  providedIn: 'root'
})

export class BookService {
  getBooks() {
    return BOOKS;
  }
}
```

6. Add the following code in the **book.component.ts** file

```
import { Component, OnInit } from '@angular/core';
import { Book } from './book';
import { BookService } from './book.service';

@Component({
  selector: 'app-book',
  templateUrl: './book.component.html',
  styleUrls: ['./book.component.css']
})
export class BookComponent implements OnInit {

  books!: Book[];

  constructor(private bookService: BookService) { }
  getBooks() {
    this.books = this.bookService.getBooks();
  }
  ngOnInit() {
    this.getBooks();
  }
}
```

7. Write the below-given code in **book.component.html**

```
<h2>My Books</h2>
<ul class="books">
  <li *ngFor="let book of books">
    <span class="badge">{{book.id}}</span> {{book.name}}
  </li>

</ul>
```

8. Add the following code in **book.component.css** which has styles for books

```
.books {
  margin: 0 0 2em 0;
  list-style-type: none;
  padding: 0;
  width: 13em;
}
.books li {
  cursor: pointer;
  position: relative;
  left: 0;
  background-color: #eee;
  margin: 0.5em;
  padding: 0.3em 0;
  height: 1.5em;
  border-radius: 4px;
}
.books li:hover {
  color: #607d8b;
  background-color: #ddd;
  left: 0.1em;
}
.books .badge {
  display: inline-block;
  font-size: small;
  color: white;
  padding: 0.8em 0.7em 0 0.7em;
  background-color: #607d8b;
  line-height: 0.5em;
  position: relative;
  left: -1px;
```

```
  top: -4px;
  height: 1.8em;
  margin-right: 0.8em;
  border-radius: 4px 0 0 4px;

}
```

9. Add the following code in app.component.html

```
<app-book></app-book>
```

10. Save the files and check the output in the browser

Output:

8.c Course Name: Angular JS
Module Name: RxJS Observables
Create and use an observable in Angular
Program:

app.component.ts

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

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

        data!: Observable<number>;
        myArray: number[] = [];
        errors!: boolean;
        finished!: boolean;

        fetchData(): void {
          this.data = new Observable(observer => {
            setTimeout(() => { observer.next(11); }, 1000),
            setTimeout(() => { observer.next(22); }, 2000),
            setTimeout(() => { observer.complete(); }, 3000);
          });
          this.data.subscribe((value) => this.myArray.push(value),
            error => this.errors = true,
            () => this.finished = true);
        }
      }
```

app.component.html

```
<b> Using Observables!</b>

<h6 style="margin-bottom: 0">VALUES:</h6>
<div *ngFor="let value of myArray">{{ value }}</div>

<div style="margin-bottom: 0">ERRORS: {{ errors }}</div>
```

```
7.
8.<div style="margin-bottom: 0">FINISHED: {{ finished }}</div>
9.
10.            <button style="margin-top: 2rem" (click)="fetchData()">Fetch Data</button>
```

output: