**SRK INSTITUTE OF TECHNOLOGY**

**Approved by AICTE, Affiliated to JNTUK, Kakinada (ISO 9001:2015 Certified Institution)**

**Department of Computer Science and Engineering**

**MEAN STACK TECHNOLOGIES-MODULE II (ANGULARJS, MONGODB)**

**LAB MANUAL**

(IV B. Tech I Sem R20 Regulation)



## Enikepadu, Vijayawada 521108

Prepared by

Ch. Pavani

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **IV Year–I Semester** |  | **L** | **T** | **P** | **C** |
| **0** | **0** | **4** | **2** |
| **MEANSTACKTECHNOLOGIES-MODULEII-ANGULARJS,MONGODB**  **(Skill Oriented Course)** | | | | | |

#### Course Outcomes:

* Build a component-based application using Angular components and enhance their functionality using directives.
* Utilize data binding for developing Angular forms and bind them with model data.
* Apply Angular built-in or custom pipes to form at the rendered data.
* Develop a single page application by using synchronous or asynchronous Angular routing.
* Make use of MongoDB queries to perform CRUD operations on document database.

#### List of Exercises:

|  |  |
| --- | --- |
| **1.a** | **CourseName:**Angular JS |
| **ModuleName:**Angular Application Setup |
| Observethelinkhttp://localhost:4200/welcomeonwhichthemCartapplicationis  running.Performthebelowactivitiestounderstandthefeaturesoftheapplication. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_24049616594198490000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **1.b** | **CourseName:**Angular JS |
| **ModuleName:**Components and Modules |
| Create a new component called hello and render Hello Angular on the page |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_28217843279641040000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **1.c** | **CourseName:**Angular JS |
| **ModuleName:**ElementsofTemplate |
| Add an event to the hello component template and when it is clicked, it should change  The course Name. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_19226434057992030000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **1.d** | **CourseName:**Angular JS |
| **ModuleName:**ChangeDetection |
| Progressively building the PoolCarzapplication |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_2560981637120771000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **2.a** | **CourseName:**Angular JS |
| **ModuleName:**StructuralDirectives -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 |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_auth\_0127637402260439042595\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |

|  |  |
| --- | --- |
| **2.b** | **CourseName:**Angular JS |
| **ModuleName:**ngFor |
| Create a courses array and rendering it in the template using ngFor directive in a listformat. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_32795774277593590000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **2.c** | **CourseName:**Angular JS |
| **ModuleName:**ngSwitch |
| Display the correct option based on the value passed to ngSwitchdirective. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_23388127475984175000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **2.d** | **CourseName:**Angular JS |
| **Module Name:**CustomStructuralDirective |
| Create a custom structural directive called 'repeat 'which should repeat the element  Given a number of times. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_24073319904331424000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **3.a** | **CourseName:**Angular JS |
| **ModuleName:**AttributeDirectives-ngStyle |
| Apply multiple CSS properties to a paragraph in a component using ngStyle. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_24037156998765367000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **3.b** | **CourseName:**Angular JS |
| **ModuleName:**ngClass |
| Apply multiple CSS classes to the text using ng Class directive. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_3459610297074182000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **3.c** | **CourseName:**Angular JS |
| **ModuleName:**CustomAttribute Directive |
| Create an attribute directive called'showMessage' which should display the given  Message in a paragraph when a user clicks on it and should change the textcolor to red. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_14783742359773809000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **4.a** | **CourseName:**Angular JS |
| **ModuleName:**PropertyBinding |
| Binding imagewithclasspropertyusingpropertybinding. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_8951964709153619000\_shared?collectionId=lex\_2085851554325460000  0\_shared&collectionType=Course |
| **4.b** | **CourseName:**Angular JS |
| **ModuleName:** Attribute Binding |
| Binding colspan attribute of a table element to the class property. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_7154252883180625000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |

|  |  |
| --- | --- |
| **4.c** | **CourseName:**Angular JS |
| **ModuleName:**Style and EventBinding |
| Binding an element using in line style and user actions like entering text in input fields. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_7417401021103822000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **5.a** | **CourseName:**Angular JS |
| **ModuleName:**BuiltinPipes |
| Display the product in lowercase and product name in uppercase using built-in  pipes. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_11810543990912035000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **5.b** | **CourseName:**Angular JS |
| **ModuleName:** PassingParameterstoPipes |
| Apply built-in pipes with parameters to display product details. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_21187073707540988000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **5.c** | **CourseName:**Angular JS |
| **ModuleName:**Nested Components Basics |
| Load Courses list Component in the root component when a userclicks on theView  Courses list button. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_24231999287700136000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **6.a** | **CourseName:**Angular JS |
| **ModuleName:**PassingdatafromContainerComponenttoChildComponent |
| Create an AppComponent that displays a dropdown with a list of courses as values in it.Create an other component called the CoursesList component and load it in  AppComponent which should display thE course details.When the users electsa course from the |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_15758356947336235000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **6.b** | **CourseName:**Angular JS |
| **ModuleName:**PassingdatafromChildComponenttoContainerComponent |
| Createan AppComponent that loads another Component called the CoursesList component.Create an other 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 th |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_2494980689916818400\_shared?collectionId=lex\_2085851554325460000  0\_shared&collectionType=Course |
| **6.c** | **CourseName:**Angular JS |
| **ModuleName:** ShadowDOM |
| Apply Shadow DOM and N on e encapsulation modes to components. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_10312243404892470000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **6.d** | **CourseName:**Angular JS |
| **ModuleName:**ComponentLifeCycle |

|  |  |
| --- | --- |
|  | Override component life-cycle hooks and logging the corresponding messages to  Understand the flow. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_10818939635948007000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **7.a** | **CourseName:**Angular JS |
| **ModuleName:** Template Driven Forms |
| Create a course registration form as a template-driven form. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_2810668513603024400\_shared?collectionId=lex\_2085851554325460000  0\_shared&collectionType=Course |
| **7.b** | **CourseName:**Angular JS |
| **ModuleName:**Model Driven Forms or Reactive Forms |
| Create an employee registration form as a reactive form. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_33704702617536004000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **7.c** | **CourseName:**Angular JS |
| **ModuleName:**CustomValidators in ReactiveForms |
| Create a custom valid atorforanemailfieldintheemployeeregistrationform(  reactive form) |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_33728128192769250000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **8.a** | **CourseName:**Angular JS |
| **ModuleName:**CustomValidatorsinTemplateDriven forms |
| Createacustomvalidator fortheemailfieldinthecourseregistrationform. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_27688491925133280000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **8.b** | **CourseName:**Angular JS |
| **ModuleName:**ServicesBasics |
| Create a Book Component which fetches book details like id, name and displays them on thepageinalistformat.Storethebookdetailsinanarray andfetchthedatausing  acustomservice. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_32584403823635940000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **8.c** | **CourseName:**Angular JS |
| **ModuleName:**RxJSObservables |
| CreateanduseanobservableinAngular. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_6209609363905256000\_shared?collectionId=lex\_2085851554325460000  0\_shared&collectionType=Course |
| **9.a** | **CourseName:**Angular JS |
| **ModuleName:**ServerCommunicationusingHttpClient |
| CreateanapplicationforServerCommunicationusingHttpClient |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_auth\_0127637395317063682615\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **9.b** | **CourseName:**Angular JS |
| **ModuleName:**CommunicatingwithdifferentbackendservicesusingAngular |

|  |  |
| --- | --- |
|  | HttpClient |
| CreateacustomservicecalledProductServiceinwhichHttpclassis usedtofetchdata  stored intheJSON files. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_4266333361795059700\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **10.**  **a** | **CourseName:**Angular JS |
| **ModuleName:** RoutingBasics, RouterLinks |
| Createmultiplecomponentsandaddroutingtoprovide navigationbetweenthem. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_3782024852517635000\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **10.**  **b** | **CourseName:**Angular JS |
| **ModuleName:**RouteGuards |
| Consideringthesameexampleusedfor routing,addrouteguardtoBooksComponent.  Only after logging in, the user should be able to access BooksComponent. If the user triestogivetheURLofBookscomponentinanothertaborwindow,oriftheusertries |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_30303325731876470000\_shared?collectionId=lex\_208585155432546000  00\_shared&collectionType=Course |
| **10.**  **c** | **CourseName:**Angular JS |
| **ModuleName:** AsynchronousRouting |
| ApplylazyloadingtoBookComponent.Iflazyloadingisnotaddedtothedemo,it  has loaded in1.14s. Observethe loadtime atthebottomofthe browserconsole. Press F12 in the browser and click the Network tab and check the Load time |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex\_9878739890118246000\_shared?collectionId=lex\_2085851554325460000  0\_shared&collectionType=Course |
| **10.**  **d** | **CourseName:**Angular JS |
| **ModuleName:**NestedRoutes |
| ImplementChildRoutestoasubmodule. |
| https://infyspringboard.onwingspan.com/web/en/viewer/web-  module/lex\_auth\_012768043900444672140\_shared?collectionId=lex\_20858515543254600000\_shared&collectionType=Course |
| **11.**  **a** | **CourseName:**MongoDBEssentials- ACompleteMongoDBGuide |
| **ModuleName:**InstallingMongoDBonthelocalcomputer,CreateMongoDBAtlas  Cluster |
| InstallMongoDBandconfigureATLAS |
| https://infyspringboard.onwingspan.com/web/en/viewer/video/lex\_auth\_01281821437  313024030083\_shared?collectionId=lex\_auth\_013177169294712832113\_shared&collectionType=Course |
| **11.**  **b** | **CourseName:**MongoDBEssentials- ACompleteMongoDBGuide |
| **ModuleName:** IntroductiontotheCRUD Operations |
| WriteMongoDBqueriestoperformCRUDoperationsondocumentusinginsert(),  find(), update(),remove() |
| https://infyspringboard.onwingspan.com/web/en/viewer/video/lex\_auth\_01281821874166169630118\_shared?collectionId=lex\_auth\_013177169294712832113\_shared&coll  ectionType=Course |
| **12.**  **a** | **CourseName:**MongoDBEssentials- ACompleteMongoDBGuide |
| **ModuleName:**CreateandDeleteDatabasesandCollections |
| WriteMongoDBqueriestoCreateanddropdatabasesandcollections. |
| https://infyspringboard.onwingspan.com/web/en/viewer/video/lex\_auth\_01281821654 |

|  |  |
| --- | --- |
|  | 119219230121\_shared?collectionId=lex\_auth\_013177169294712832113\_shared&coll  ectionType=Course |
| **12.**  **b** | **CourseName:**MongoDBEssentials- ACompleteMongoDBGuide |
| **ModuleName:**IntroductiontoMongoDB Queries |
| WriteMongoDBqueriestoworkwithrecordsusingfind(),limit(),sort(),  createIndex(), aggregate(). |
| https://infyspringboard.onwingspan.com/web/en/viewer/video/lex\_auth\_0132890816264519682505\_shared?collectionId=lex\_auth\_013177169294712832113\_shared&colle  ctionType=Course |

**TextBooks:**

1. ProgrammingtheWorldWideWeb,7thEdition, RobetWSebesta, Pearson.
2. ProMeanStackDevelopment,1stEdition,ELadElrom,ApressO’Reilly.
3. FullStackJavaScript Development withMEAN,ColinJIhrig,AdamBretz,1st edition,SitePoint, SitePoint Pty. Ltd., O'Reilly Media.
4. MongoDB–TheDefinitiveGuide,2ndEdition,KristinaChodorow, O’Reilly

#### Softwareconfigurationandinstallation:

1. Angular

Setupdetails:Angular ApplicationSetup-Internal-ViewerPage|InfosysSpringboard (onwingspan.com)

1. MongoDB

TOC- MongoDBEssentials- ACompleteMongoDBGuide|InfosysSpringboard(onwingspan.com)

#### WebLinks:

* 1. https://infyspringboard.onwingspan.com/en/app/toc/lex\_20858515543254600000\_shared/overview(Angular JS)
  2. https://infyspringboard.onwingspan.com/en/app/toc/lex\_auth\_013177169294712832113\_shared/overview(MongoDB)

**EXERCISE-1**

**1a.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.

| **REQUIREMENTS** | **DETAILS** |
| --- | --- |
| Node.js | Angular requires an [active LTS or maintenance LTS](https://nodejs.org/about/releases) version of Node.js.  For information see the [version compatibility](https://angular.io/guide/versions) guide.  For more information on installing Node.js, see [nodejs.org](https://nodejs.org/). If you are unsure what version of Node.js runs on your system, run node -v in a terminal window. |
| npm package manager | Angular, the Angular CLI, and Angular applications depend on [npm packages](https://docs.npmjs.com/getting-started/what-is-npm) for many features and functions. To download and install npm packages, you need an npm package manager. This guide uses the [npm client](https://docs.npmjs.com/cli/install) command line interface, which is installed with Node.js by default. To check that you have the npm client installed, run npm -v in a terminal window. |

**INSTALLING ANGULAR.JS IN VISUALSTUDIO**

Install the Angular CLI

You can use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

To install the Angular CLI, open a terminal window and run the following command:

**npm install -g @angular/cli**

On Windows client computers, the execution of PowerShell scripts is disabled by default. To allow the execution of PowerShell scripts, which is needed for npm global binaries, you must set the following [execution policy](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies):

**Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned**

Carefully read the message displayed after executing the command and follow the instructions. Make sure you understand the implications of setting an execution policy.

Create a workspace and initial application

You develop apps in the context of an Angular [workspace](https://angular.io/guide/glossary#workspace).

To create a new workspace and initial starter app:

1. Run the CLI command ng new and provide the name my-app, as shown here:

content\_copyng new my-app

1. The ng new command prompts you for information about features to include in the initial app. Accept the defaults by pressing the Enter or Return key.

The Angular CLI installs the necessary Angular npm packages and other dependencies. This can take a few minutes.

The CLI creates a new workspace and a simple Welcome app, ready to run.

Run the application

The Angular CLI includes a server, for you to build and serve your app locally.

1. Navigate to the workspace folder, such as my-app.
2. Run the following command:

**my-app**

**ng serve --open**

The ng serve command launches the server, watches your files, and rebuilds the app as you make changes to those files.

The --open (or just -o) option automatically opens your browser to http://localhost:4200/.

**Output:**

PS C:\Users\SRK\Desktop\cseb rockers> **npm install -g @angular/cli**

PS C:\Users\SRK\Desktop\cseb rockers> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

PS C:\Users\SRK\Desktop\cseb rockers> ng new my-app

Node.js version v19.8.1 detected.

Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/releases/.

? Would you like to add Angular routing? Yes

? Which stylesheet format would you like to use? Less [ http://lesscss.org

]

CREATE my-app/angular.json (2874 bytes)

CREATE my-app/package.json (1037 bytes)

CREATE my-app/README.md (1059 bytes)

CREATE my-app/tsconfig.json (901 bytes)

CREATE my-app/.editorconfig (274 bytes)

CREATE my-app/.gitignore (548 bytes)

CREATE my-app/tsconfig.app.json (263 bytes)

CREATE my-app/tsconfig.spec.json (273 bytes)

CREATE my-app/.vscode/extensions.json (130 bytes)

CREATE my-app/.vscode/launch.json (470 bytes)

CREATE my-app/.vscode/tasks.json (938 bytes)

CREATE my-app/src/favicon.ico (948 bytes)

CREATE my-app/src/index.html (291 bytes)

CREATE my-app/src/styles.less (80 bytes)

CREATE my-app/src/app/app-routing.module.ts (245 bytes)

CREATE my-app/src/app/app.module.ts (393 bytes)

CREATE my-app/src/app/app.component.html (23115 bytes)

CREATE my-app/src/app/app.component.spec.ts (991 bytes)

CREATE my-app/src/app/app.component.ts (211 bytes)

CREATE my-app/src/app/app.component.less (0 bytes)

CREATE my-app/src/assets/.gitkeep (0 bytes)

✔ Packages installed successfully.

Directory is already under version control. Skipping initialization of git.

PS C:\Users\SRK\Desktop\cseb rockers> cd my-app

>> ng serve --open

Node.js version v19.8.1 detected.

Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/releases/.

? Would you like to share pseudonymous usage data about this project with the Angular Team

at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more

details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following

command will disable this feature entirely:

ng analytics disable

Global setting: enabled

Local setting: enabled

Effective status: enabled

✔ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size

vendor.js | vendor | 2.28 MB |

polyfills.js | polyfills | 333.14 kB |

styles.css, styles.js | styles | 230.67 kB |

main.js | main | 48.08 kB |

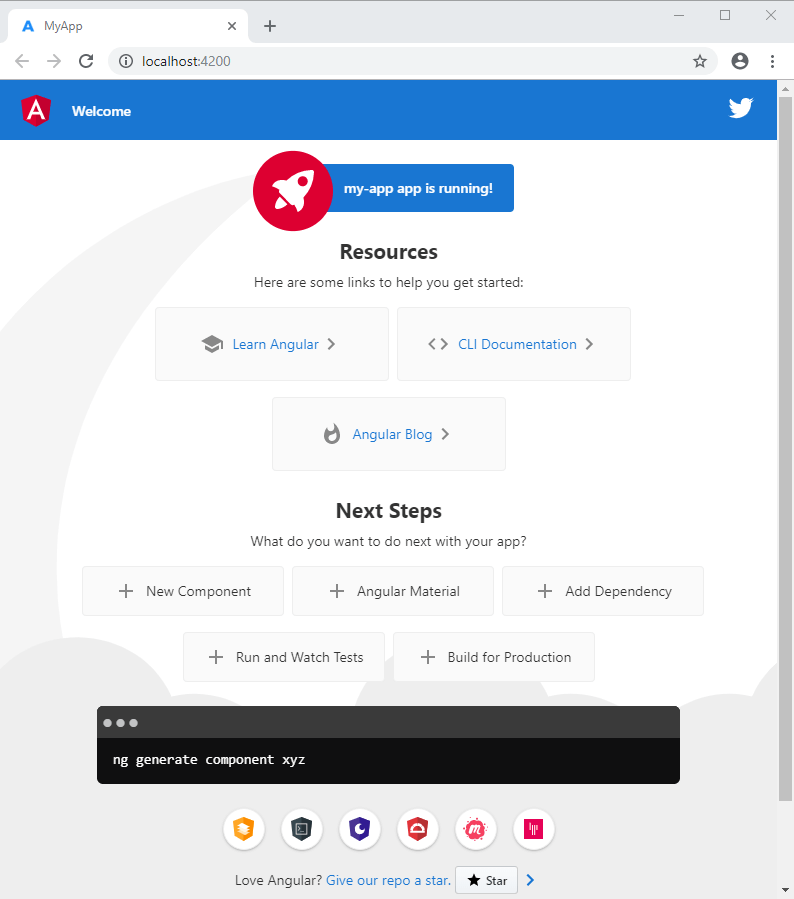
runtime.js | runtime | 6.51 kB |

| Initial Total | 2.89 MB

Build at: 2023-07-24T04:42:02.409Z - Hash: 8542936140e45a51 - Time: 7980ms

\*\* Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ \*\*

√ Compiled successfully.



**1b. Course Name: Angular JS**

**Module Name: Components and Modules Create a new component called hello and render Hello Angular on the page.**

**Process :**

**Step 1** - Test the default app

In this step, after you download the default starting app, you build the default Angular app. This confirms that your development environment has what you need to continue the tutorial.

In the Terminal pane of your IDE:

1. In your project directory, navigate to the first-app directory.
2. Run this command to install the dependencies needed to run the app.

**npm install**

1. Run this command to build and serve the default app.

**ng serve**

The app should build without errors.

1. In a web browser on your development computer, open http://localhost:4200.
2. Confirm that the default web site appears in the browser.
3. You can leave ng serve running as you complete the next steps.

**Step 2** - Review the files in the project

In this step, you get to know the files that make up a default Angular app.

In the Explorer pane of your IDE:

1. In your project directory, navigate to the first-app directory.

* Open the src directory to see these files.
* In the file explorer, find the Angular app files (/src).
* index.html is the app's top level HTML template.
* style.css is the app's top level style sheet.
* main.ts is where the app start running.
* favicon.ico is the app's icon, just as you would find in any web site.
* In the file explorer, find the Angular app's component files (/app).
* app.component.ts is the source file that describes the app-root component. This is the top-level Angular component in the app. A component is the basic building block of an Angular application. The component description includes the component's code, HTML template, and styles, which can be described in this file, or in separate files.

In this app, the styles are in a separate file while the component's code and HTML template are in this file.

* app.component.css is the style sheet for this component.
* New components are added to this directory.
* In the file explorer, find the image directory (/assets) that contains images used by the app.
* In the file explorer, find the files and directories that an Angular app needs to build and run, but they are not files that you normally interact with.
* .angular has files required to build the Angular app.
* .e2e has files used to test the app.
* .node\_modules has the node.js packages that the app uses.
* angular.json describes the Angular app to the app building tools.
* package.json is used by npm (the node package manager) to run the finished app.
* tsconfig.\* are the files that describe the app's configuration to the TypeScript compiler.

After you have reviewed the files that make up an Angular app project, continue to the next step.

**Step 3** - Create Hello World

In this step, you update the Angular project files to change the displayed content.

In your IDE:

1. Open first-app/src/index.html.
2. In index.html, replace the <title> element with this code to update the title of the app.

Replace in src/index.html

**<title>Homes</title>**

Then, save the changes you just made to index.html.

1. Next, open first-app/src/app/app.component.ts.
2. In app.component.ts, in the @[Component](https://angular.io/api/core/Component) definition, replace the template line with this code to change the text in the app component.

Replace in src/app/app.component.ts

**template: `<h1>Hello world!</h1>`,**

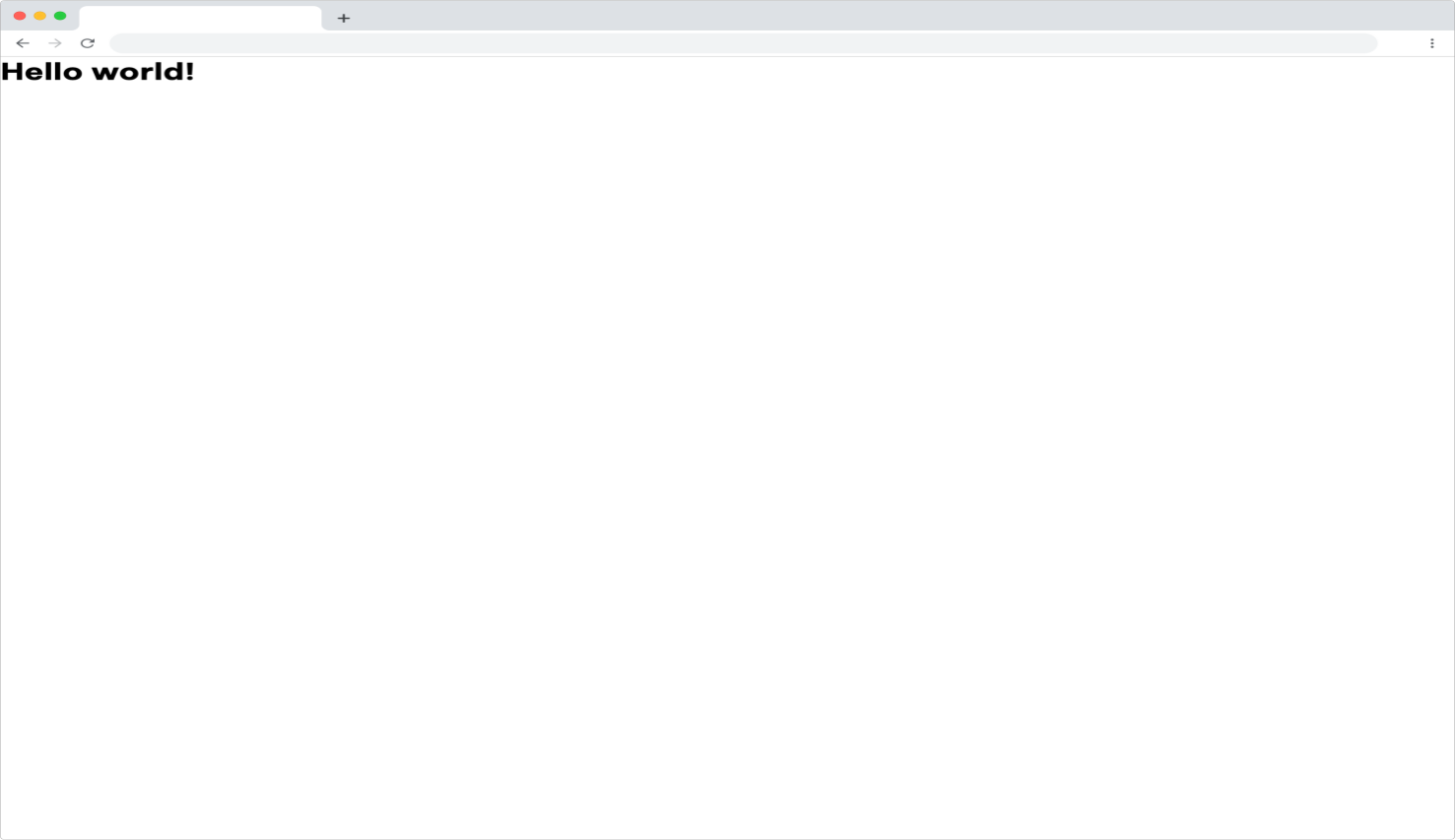
1. In app.component.ts, in the AppComponent class definition, replace the title line with this code to change the component title.

Replace in src/app/app.component.ts

**title = 'homes';**

Then, save the changes you made to app.component.ts.

1. If you stopped the ng serve command from step 1, in the Terminal window of your IDE, run ng serve again.
2. Open your browser and navigate to localhost:4200 and confirm that the app builds without error and displays *Hello world* in the title and body of your app:



**Program:**

**Helloworld.js**

**Index.html:**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Default</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="preconnect" href="https://fonts.googleapis.com">

  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

 <link href="https://fonts.googleapis.com/css2?family=Be+Vietnam+Pro:ital,wght@0,400;0,700;1,400;1,700&display=swap" rel="stylesheet">

</head>

<body>

  <app-root></app-root>

</body>

</html>

**Main.ts:**

import { bootstrapApplication,provideProtractorTestingSupport } from '@angular/platform-browser';

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

bootstrapApplication(AppComponent,

    {providers: [provideProtractorTestingSupport()]})

  .catch(err => console.error(err));

**Style.css:**

 {

  margin: 0;

  padding: 0;

}

body {

  font-family: 'Be Vietnam Pro', sans-serif;

}

:root {

  --primary-color: #605DC8;

  --secondary-color: #8B89E6;

  --accent-color: #e8e7fa;

  --shadow-color: #E8E8E8;

}

button.primary {

  padding: 10px;

  border: solid 1px var(--primary-color);

  background: var(--primary-color);

  color: white;

  border-radius: 8px;

}

**App.component.css**

:host {

  --content-padding: 10px;

}

header {

  display: block;

  height: 60px;

  padding: var(--content-padding);

  box-shadow: 0px 5px 25px var(--shadow-color);

}

.content {

  padding: var(--content-padding);

}

**App.component.ts**

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

@Component({

  selector: 'app-root',

  standalone: true,

  imports: [],

  template: `<h1>Hello</h1>`,

  styleUrls: ['./app.component.css'],

})

export class AppComponent {

  title = 'default';

}

**Output:**

~/projects/qqsjdl--run

❯ npm install --legacy-peer-deps && npm start

npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

npm WARN deprecated har-validator@5.1.5: this library is no longer supported

npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

npm WARN deprecated protractor@7.0.0: We have news to share - Protractor is deprecated and will reach end-of-life by Summer 2023. To learn more and find out about other options please refer to this post on the Angular blog. Thank you for using and contributing to Protractor. https://goo.gle/state-of-e2e-in-angular

added 1062 packages in 10s

104 packages are looking for funding

run `npm fund` for details

> angular.io-example@0.0.0 start

> ng serve

✔ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size

vendor.js | vendor | 1.98 MB |

polyfills.js | polyfills | 333.17 kB |

styles.css, styles.js | styles | 231.00 kB |

runtime.js | runtime | 6.53 kB |

main.js | main | 4.11 kB |

| Initial Total | 2.54 MB

Build at: 2023-07-24T06:01:39.536Z - Hash: 12a28dce4fb5ef2b - Time: 6366ms

\*\* Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ \*\*

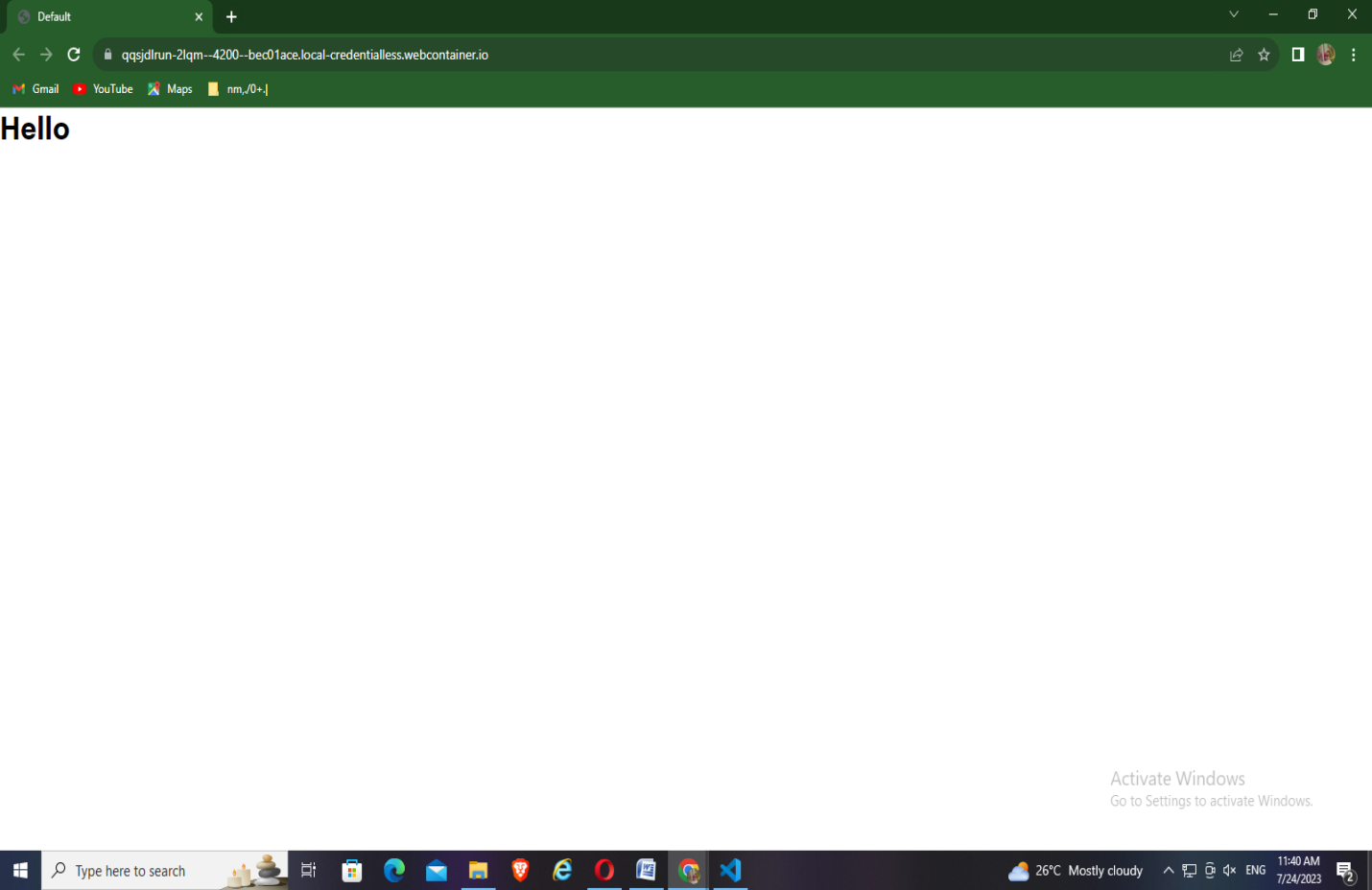
✔ Compiled successfully.

✔ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2023-07-24T06:01:40.019Z - Hash: 12a28dce4fb5ef2b - Time: 377ms

✔ Compiled successfully.



**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 course Name.

**hello.component.ts:**

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

@Component({

  selector: 'app-hello',

  templateUrl: "./hello.component.html",

  styleUrls: ['./hello.component.css']

})

export class HelloComponent implements OnInit {

  courseName = "MSD";

  constructor() { }

  ngOnInit() {

  }

  changeName() {

    this.courseName = "CSE";

  }

}

**hello.component.html:**

<h1>Welcome</h1>

<h2>Course Name: {{ courseName }}</h2>

<button><p (click)="changeName()">Click here to change</p></button>

**hello.component.css:**

p {

    color:rgb(255, 60, 0);

font-size:20px;

}

**Output:**

PS C:\Users\CSE\Desktop\angular\MyApp\src\app> ng serve --open --port 3000

✔ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size

vendor.js | vendor | 2.28 MB |

polyfills.js | polyfills | 333.15 kB |

styles.css, styles.js | styles | 230.44 kB |

main.js | main | 8.79 kB |

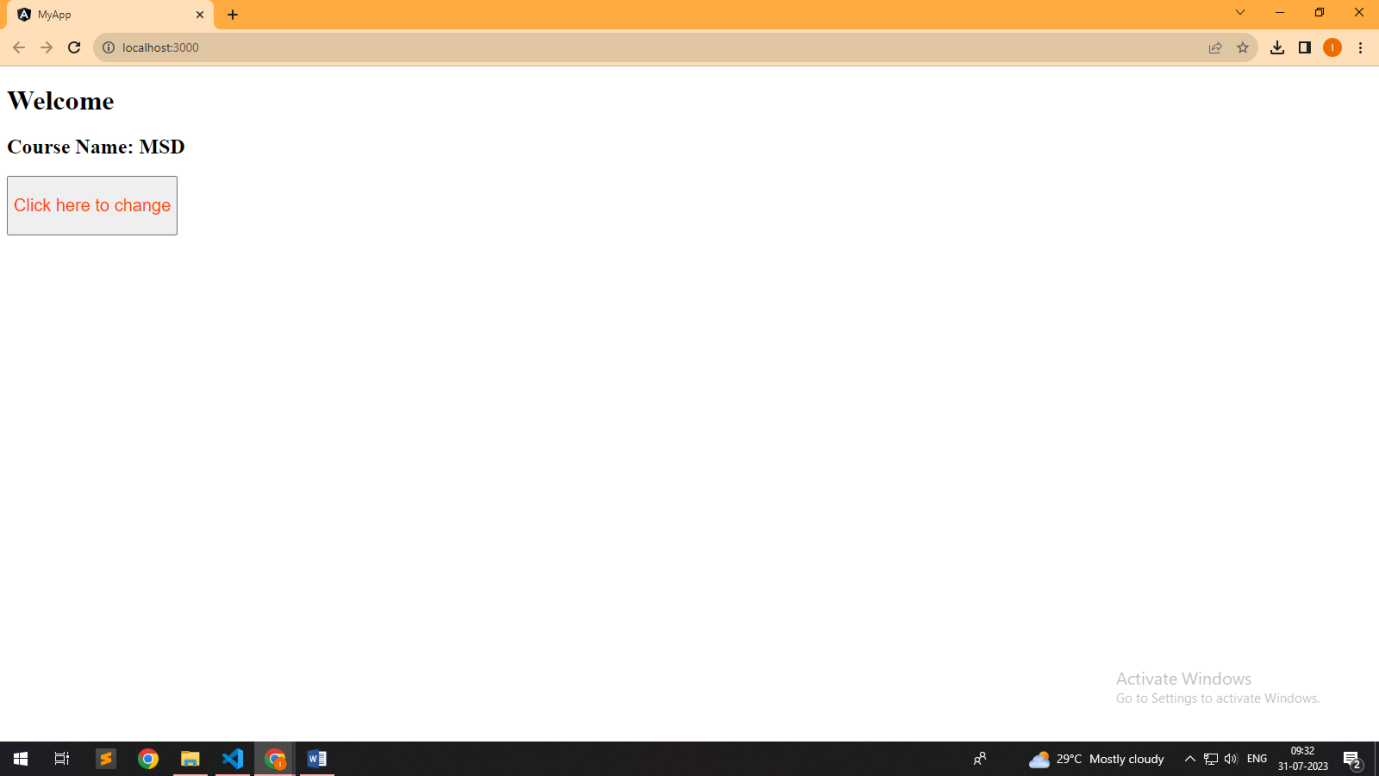
runtime.js | runtime | 6.51 kB |

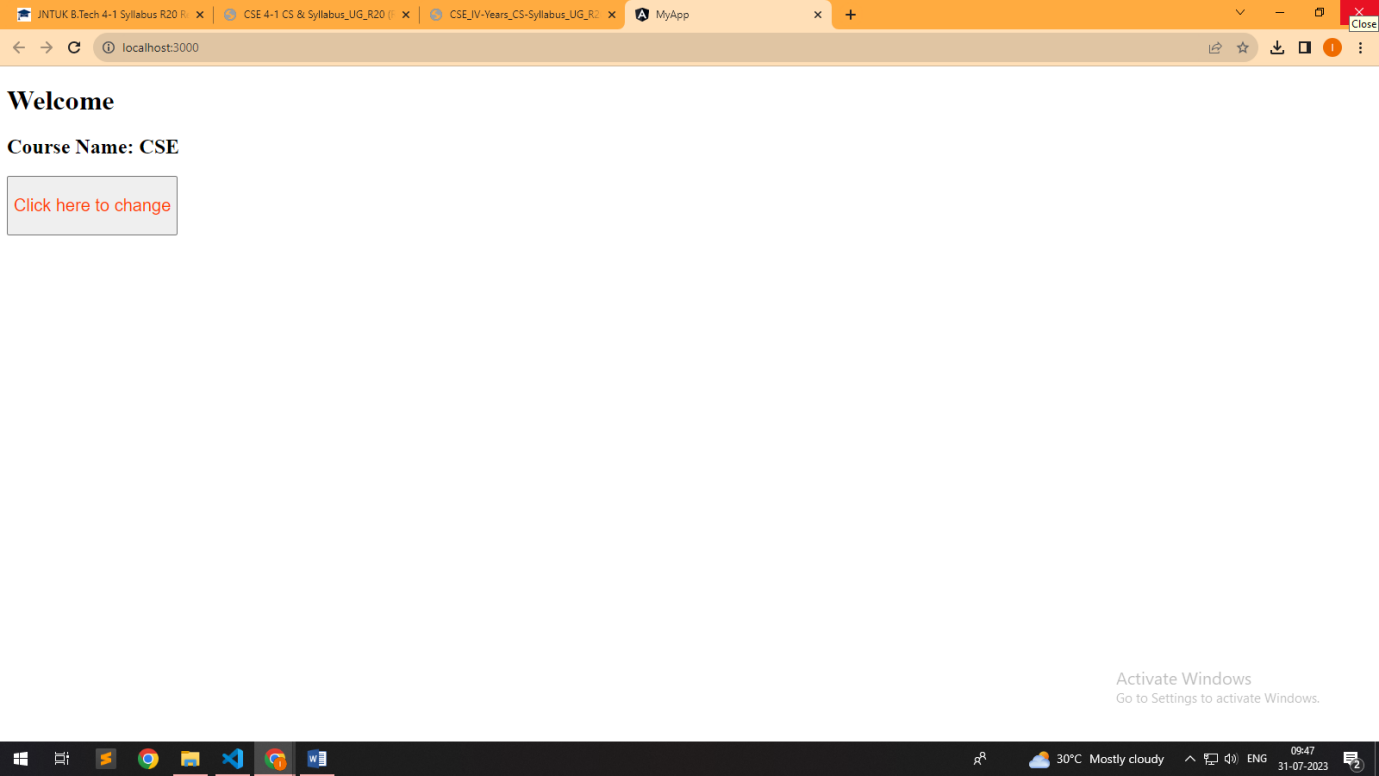
| Initial Total | 2.85 MB

Build at: 2023-07-31T03:54:06.304Z - Hash: 538059ec4cb5376e - Time: 12617ms

\*\* Angular Live Development Server is listening on localhost:3000, open your browser on http://localhost:3000/ \*\*

√ Compiled successfully.





**1.d. Course Name: Angular JS**

**Module Name: Change Detection**

**progressively building the PoolCarz application**

**Change detection** is the process through which Angular checks to see whether your application state has changed, and if any DOM needs to be updated. At a high level, Angular walks your components from top to bottom, looking for changes. Angular runs its change detection mechanism periodically so that changes to the data model are reflected in an application’s view. Change detection can be triggered either manually or through an asynchronous event (for example, a user interaction or an XMLHttpRequest completion).

Change detection is highly optimized and performant, but it can still cause slowdowns if the application runs it too frequently.

In this guide, you’ll learn how to control and optimize the change detection mechanism by skipping parts of your application and running change detection only when necessary.

The Angular change detection mechanism is much more transparent and easier to reason about than its equivalent in AngularJs. But there are still situations (like when doing performance optimizations) when we really need to know what's going on under the hood. So let's dig deeper into change detection by going over the following topics:

How is change detection implemented?

What does an Angular change detector look like, can I see it?

How does the default change detection mechanism work

turning on/off change detection, and triggering it manually

Avoiding change detection loops: Production vs Development mode

What does the OnPush change detection mode actually do?

Using Immutable.js to simplify the building of Angular apps

**Overriding browser default mechanisms**

What happens is that Angular at startup time will patch several low-level browser APIs, such as for example addEventListener, which is the browser function used to register all browser events, including click handlers. Angular will replace addEventListener with a new version that does the equivalent of this:

|  |  |
| --- | --- |
|  | // this is the new version of addEventListener |
|  | function addEventListener(eventName, callback) { |
|  | // call the real addEventListener |
|  | callRealAddEventListener(eventName, function() { |
|  | // first call the original callback |
|  | callback(...); |
|  | // and then run Angular-specific functionality |
|  | var changed = angular.runChangeDetection(); |
|  | if (changed) { |
|  | angular.reRenderUIPart(); |
|  | } |
|  | }); |
|  | } |

The change detection tree

|  |  |  |
| --- | --- | --- |
|  | @Component({ |  |
|  | selector: 'todo-item', |  |
|  | template: `<span class="todo noselect" |  |
|  | (click)="onToggle()">{{todo.owner.firstname}} - {{todo.description}} |  |
|  | - completed: {{todo.completed}}</span>` |  |
|  | }) |  |
|  | export class TodoItem { |  |
|  | @Input() |  |
|  | todo:Todo; |  |
|  |  |  |
|  | @Output() |  |
|  | toggle = new EventEmitter<Object>(); |  |
|  |  |  |
|  | onToggle() { |  |
|  | this.toggle.emit(this.todo); |  |
|  | } |  |
|  | } |  |

Each Angular component has an associated change detector, which is created at application startup time. For example, take the following  
TodoItem component:

**Program:**

**Index.html:**

<!Doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>PoolCarz</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <a href="https://m.media-amazon.com/images/I/51hoKtF9B2L.jpg">LOKESH</a>

</head>

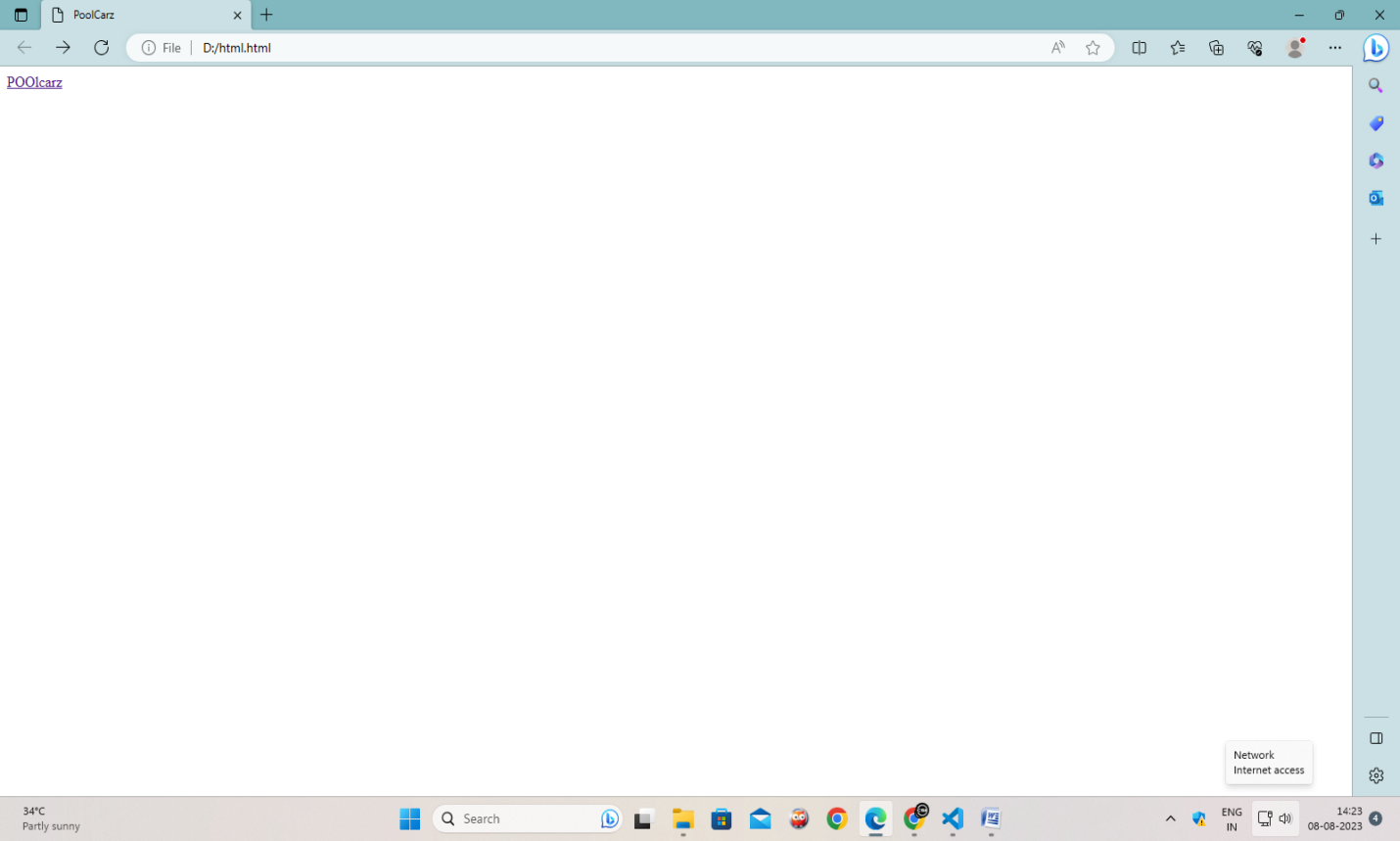
<body>

  <app-root></app-root>

</body>

</html>

**Output:**



**EXERCISE-2**

**Course Name: Angular JS**

1. 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 <>" message otherwise it should render "Invalid Login!!! Please try again..." message https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_auth\_0127637402260439042595\_shared?collectionId=lex\_2085851554

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>

**App.component.ts**

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;

    }

  }

}

**App.module.ts**

import { NgModule } from '@angular/core';

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

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

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

import { AuthorsComponent } from './authors.component';

@NgModule({

  imports: [BrowserModule, FormsModule,

    HttpClientModule],

  declarations: [AppComponent, AuthorsComponent],

  bootstrap: [AppComponent]

})

export class AppModule { }

**index.html**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>HelloWorld</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

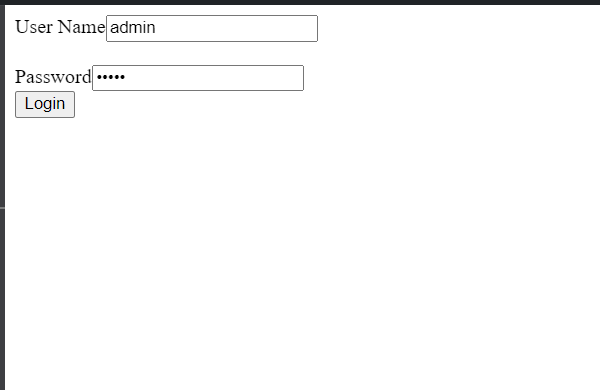
<body>

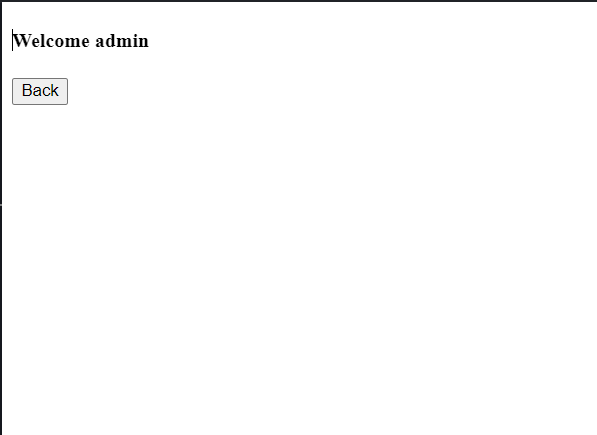
  <app-root></app-root>

</body>

</html>

**Output:**





Top of Form





2b) **Course Name**: Angular JS Module Name: ngFor Create a courses array and rendering it in the template using ngFor directive in a list format. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_32795774277593590000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

**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' }

  ];

}

**App.component.html**

<ul>

  <li \*ngFor="let course of courses; let i = index">

    {{ i }} - {{ course.name }}

  </li>

</ul>

**Index.html**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>HelloWorld</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

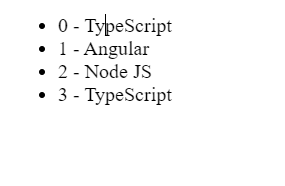
<body>

  <app-root></app-root>

</body>

</html>

**Output:**



2c) Course Name: Angular JS Module Name: ngSwitch Display the correct option based on the value passed to ngSwitch directive. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_23388127475984175000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

**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++;

  }

}

**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>

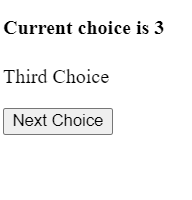
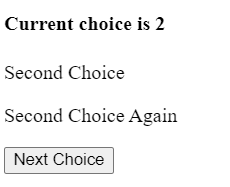
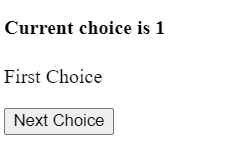
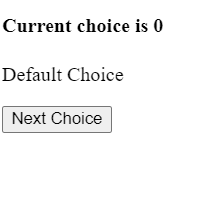
  <button (click)="nextChoice()">

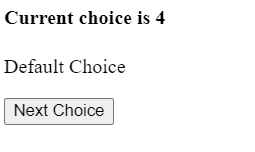
                Next Choice

        </button>

</div>

**Output:**





2d) **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. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_24073319904331424000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

**App.module.ts**

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

import { NgModule } from '@angular/core';

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

import { RepeatDirective } from './repeat.directive';

@NgModule({

  declarations: [

    AppComponent,

    RepeatDirective

  ],

  imports: [

    BrowserModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

repeat.directive.ts

import { Directive, TemplateRef, ViewContainerRef, Input } from '@angular/core';

@Directive({

  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);

    }

  }

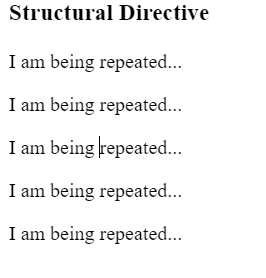
}

**App.component.html**

<h3>Structural Directive</h3>

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

**Output:**



\

**EXERCISE-3**

3a) Course Name: Angular JS Module Name: Attribute Directives - ngStyle Apply multiple CSS properties to a paragraph in a component using ngStyle. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_24037156998765367000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

**App.component.html**

<p [ngStyle]="{

  color:colorName,

  'font-weight':fontWeight,

  borderBottom: borderStyle

}">

Demo for attribute directive ngStyle

</p>

**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';

}

**App.component.css**

p {

font-family: Lato;

}

**App.module.ts**

import { NgModule } from '@angular/core';

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

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

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

import { AuthorsComponent } from './authors.component';

@NgModule({

  imports: [BrowserModule, FormsModule,

    HttpClientModule],

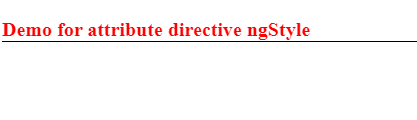
  declarations: [AppComponent, AuthorsComponent],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Output:**



3b) **Course Name: Angular JS**

Module Name: ngClass Apply multiple CSS classes to the text using ngClass directive. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_3459610297074182000\_shared?collectionId=lex\_2085851554325460000 0\_shared&collectionType=Course

**App.component.css**

.bordered {

border: 1px dashed black;

background-color: #eee;

}

.color {

color: blue;

}

**App.component.htm**l

<div [ngClass]="{bordered: isBordered, color: isColor}">

  Border {{ isBordered ? "ON" : "OFF" }}

</div>

App.component.ts

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

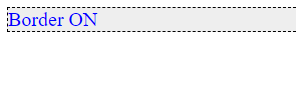
export class AppComponent  {

   isBordered = true;

   isColor = true;

}

**Output:**



3c) Course Name: Angular JS Module Name: Custom Attribute Directive Create an attribute directive called 'showMessage' which should display the given message in a paragraph when a user clicks on it and should change the text color to red. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_14783742359773809000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

**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';

}

**Message.directive.ts**

import { Directive, ElementRef, Renderer2, HostListener, Input } from '@angular/core';

@Directive({

  selector: '[appMessage]',

})

export class MessageDirective {

  @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');

  }

}

**App.component.html**

<h3>Attribute Directive</h3>

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

**App.component.css**

h3 {

  color: #369;

  font-family: Arial, Helvetica, sans-serif;

  font-size: 250%;

}

p {

  color: #ff0080;

  font-family: Arial, Helvetica, sans-serif;

  font-size: 150%;

}

**App.component.ts**

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

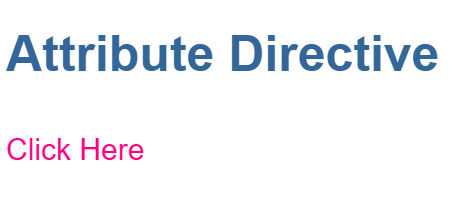
})

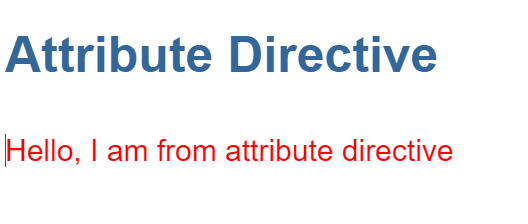
export class AppComponent {

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

}

**Output**





**EXERCISE-4**

4**a) Course Name: Angular JS**

Module Name: Property Binding Binding image with class property using property binding. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_8951964709153619000\_shared?collectionId=lex\_2085851554325460000 0\_shared&collectionType=Course

Code:

**App.components.ts:**

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

@Component({

  selector: 'app-root',

    template: `<div>

                    <img [src] = 'ImagePath' />

               </div>`

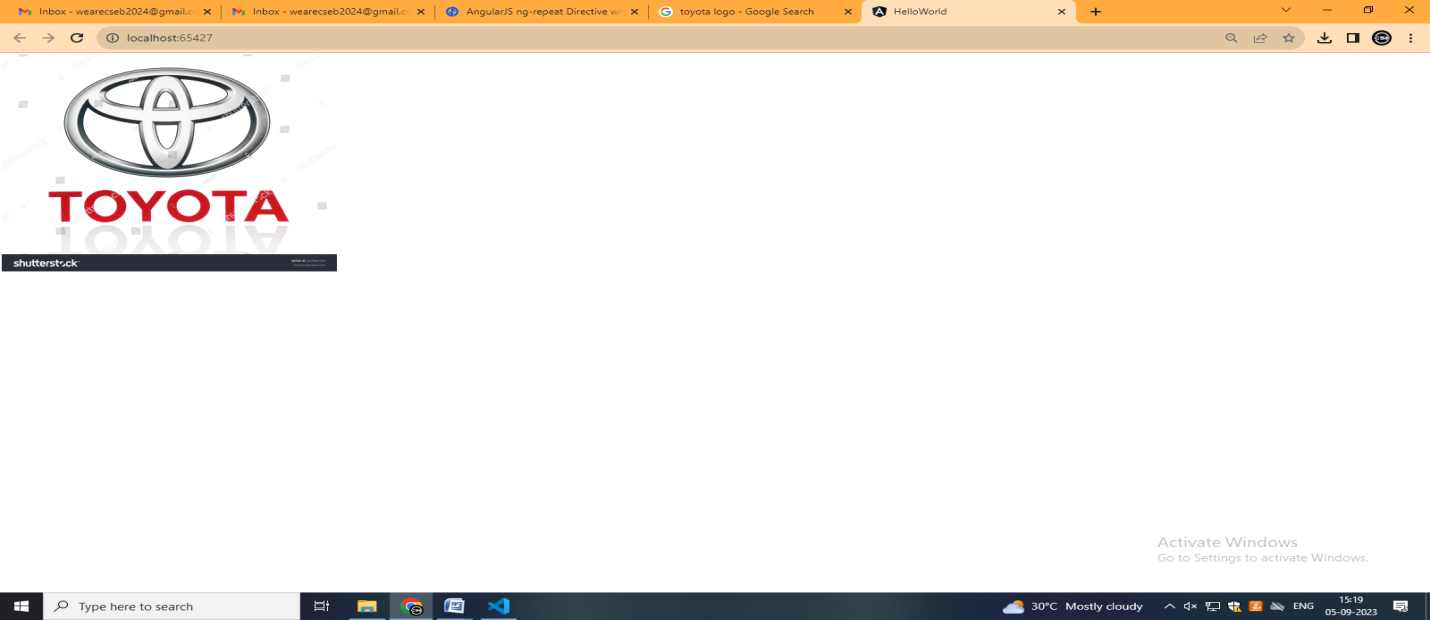
})

export class AppComponent {

  ImagePath : string = "https://upload.wikimedia.org/wikipedia/commons/b/b1/Tata\_Consultancy\_Services\_Logo.svg";

}

**Output:**



4b) **Course Name: Angular JS Module Name: Attribute Bindin**g

Binding colspan attribute of a table element to the class property. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_7154252883180625000\_shared?collectionId=lex\_2085851554325460000 0\_shared&collectionType=Course

**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';

}

**App.component.html**

<table border=1>

  <tr>

      <td [attr.colspan]="colspanValue"> First </td>

      <td>Second</td>

  </tr>

  <tr>

      <td>Third</td>

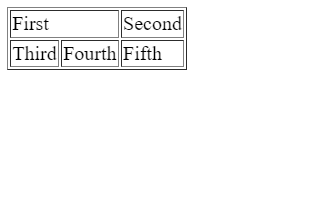
      <td>Fourth</td>

      <td>Fifth</td>

  </tr>

</table>

**Output:**



4c) **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. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_7417401021103822000\_shared?collectionId=lex\_2085851554325460000 0\_shared&collectionType=Course

**Syntax for style binding**

[style.styleproperty]

Example:

For inline style

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

The ngStyle directive is preferred when it is required to set multiple inline styles at the same time.

Event binding

<button on-click = "onSubmit(username.value,password.value)">Login</button>

Output:



**EXERCISE-5**

5a)**Course Name: Angular JS Module Name: Built in Pipes**

Display the product code in lowercase and product name in uppercase using built-in pipes. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_11810543990912035000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

uppercase

This pipe converts the template expression into uppercase.

Syntax:

{{ expression | uppercase }}

Example:

{{ "Laptop" | uppercase }}

Output:

LAPTOP

lowercase

This pipe converts the template expression into lowercase.

Syntax:

{{ expression | lowercase }}

Example:

{{ "LAPTOP" | lowercase }}

Output:

laptop

titlecase

This pipe converts the first character in each word of the given expression into a capital letter.

Syntax:

{{ expression | titlecase }}

Example:

{{ "product details" | titlecase }}

**Output:**

Product Details

**Program:**

**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 = 'Laptop';

}

**App.component.html**

<h3> {{ title | titlecase}} </h3>

<table style="text-align:left">

<tr>

<th> Product Code </th>

<td> {{ productCode | lowercase }} </td>

</tr>

<tr>

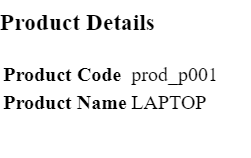
<th> Product Name </th>

<td> {{ productName | uppercase }} </td>

</tr>

</table>

**Output:**



5b) **Course Name: Angular JS Module Name: Passing Parameters to Pipes**

Apply built-in pipes with parameters to display product details. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_21187073707540988000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

A pipe can also have optional parameters to change the output. To pass parameters, after a pipe name add a colon( : ) followed by the parameter value.

Syntax:

pipename : parametervalue

A pipe can also have multiple parameters as shown below

pipename : parametervalue1:parametervalue2

Below are the built-in pipes present in Angular, which accept optional parameters using which the pipe's output can be fine-tuned.

currency

This pipe displays a currency symbol before the expression. By default, it displays the currency symbol $

Syntax:

{{ expression | currency:currencyCode:symbol:digitInfo:locale }}

currencyCode is the code to display such as INR for the rupee, EUR for the euro, etc.

symbol is a Boolean value that represents whether to display currency symbol or code.

code: displays code instead of a symbol such as USD, EUR, etc.

symbol (default): displays symbol such as $ etc.

symbol-narrow: displays the narrow symbol of currency. Some countries have two symbols for their currency, regular and narrow. For example, the Canadian Dollar CAD has the symbol as CA$ and symbol-narrow as $.

digitInfo is a string in the following format

{minIntegerDigits}.{minFractionDigits} - {maxFractionDigits}

minIntegerDigits is the minimum integer digits to display. The default value is 1

minFractionDigits is the minimum number of digits to display after the fraction. The default value is 0

maxFractionDigits is the maximum number of digits to display after the fraction. The default value is 3

locale is used to set the format followed by a country/language. To use a locale,  the locale needs to be registered in the root module.

For Example,to set locale to French (fr), add the below statements in app.module.ts

import { registerLocaleData } from '@angular/common';

import localeFrench from '@angular/common/locales/fr';

registerLocaleData(localeFrench);

Examples:

{{ 25000 | currency }} will display $25,000.00

{{ 25000 | currency:'CAD' }} will display CA$25,000.00

{{ 25000 | currency:'CAD':'code' }} will display CAD25,000.00

{{ 25000 | currency:'CAD':'symbol':'6.2-3'}} will display CA$025,000.00

{{ 25000 | currency:'CAD': 'symbol-narrow':'1.3'}} will display $25,000.000

{{ 250000 | currency:'CAD':'symbol':'6.3'}} will display CA$250,000.000

{{ 250000 | currency:'CAD':'symbol':'6.3':'fr'}} will display 250 000,000 CA$

**date**

This pipe can be used to display the date in the required format

Syntax:

{{ expression | date:format:timezone:locale }}

An expression is a date or number in milliseconds

The format indicates in which form the date/time should be displayed. Following are the pre-defined options for it.

'medium' :equivalent to 'MMM d, y, h:mm:ss a' (e.g. Jan 31, 2018, 11:05:04 AM)

'short': equivalent to 'M/d/yy, h:mm a' (e.g. 1/31/2018, 11:05 AM)

'long': equivalent to 'MMMM d, y, h:mm:ss a z' (e.g. January 31, 2018 at 11:05:04 AM GMT+5)

'full': equivalent to 'EEEE, MMMM d, y, h:mm:ss a zzzz' (e.g. Wednesday, January 31, 2018 at 11:05:04 AM GMT+05:30)

'fullDate' : equivalent to 'EEEE, MMMM d, y' (e.g. Wednesday, January 31, 2018)

'longDate' : equivalent to 'MMMM d, y' (e.g. January 31, 2018)

'mediumDate' : equivalent to 'MMM d, y' (e.g. Jan 31, 2018)

'shortDate' : equivalent to 'M/d/yy' (e.g. 1/31/18)

'mediumTime' : equivalent to 'h:mm:ss a' (e.g. 11:05:04 AM)

'shortTime' :  equivalent to 'h:mm a' (e.g. 11:05 AM)

'longTime': equivalent to 'h:mm a' (e.g. 11:05:04 AM GMT+5)

'fullTime': equivalent to 'h:mm:ss a zzzz' (e.g. 11:05:04 AM GMT+05:30)

Timezone to be used for formatting. For example, ’+0430’ (4 hours, 30 minutes east of the Greenwich meridian) If not specified, the local system timezone of the end-user's browser will be used.

locale is used to set the format followed by a country/language. To use a locale, register the locale in the root module.

For Example, to set locale to French (fr), add the below statements in app.module.ts

import { registerLocaleData } from '@angular/common';

import localeFrench from '@angular/common/locales/fr';

registerLocaleData(localeFrench);

Examples:

{{ "6/2/2017" | date }} will display Jun 2, 2017

{{ "6/2/2017, 11:30:45 AM" | date:'medium' }} will display Jun 2, 2017, 11:30:45 AM

{{ "6/2/2017, 11:30:45 AM" | date:'mmss' }} will display 3045

{{"1/31/2018, 11:05:04 AM" | date:'fullDate':'0':'fr'}} will display mercredi 31 janvier 2018

{{ 90000000 | date }} will display Jan 2, 1970 – date pipe will start from Jan 1, 1970 and based on the given number of milliseconds, it displays the datepercent

This pipe can be used to display the number as a percentage

Syntax:

{{ expression | percent:digitInfo:locale }}

digitInfo is a string in the following format

{minIntegerDigits}.{minFractionDigits} - {maxFractionDigits}

minIntegerDigits is the minimum integer digits to display. The default value is 1

minFractionDigits is the minimum number of digits to display after the fraction. The default value is 0.

maxFractionDigits is the maximum number of digits to display after the fraction. The default value is 3.

locale is used to set the format followed by a country/language. To use a locale,  register the locale in the root module.

For Example, to set locale to French (fr), add the below statements in app.module.ts

import { registerLocaleData } from '@angular/common';

import localeFrench from '@angular/common/locales/fr';

registerLocaleData(localeFrench);

Examples:

{{ 0.1 | percent }} will display 10%

{{ 0.1 | percent:'2.2-3' }} will display 10.00%

{{ 0.1 | percent:'2.2-3': 'fr' }} will display 10.00 %

slice

This pipe can be used to extract a subset of elements or characters from an array or string respectively.

Syntax:

{{ expression | slice:start:end }}

The expression can be an array or string

start represents the starting position in an array or string to extract items. It can be a

positive integer which will extract from the given position till the end

negative integer which will extract the given number of items from the end

end represents the ending position in an array or string for extracting items. It can be

positive number that returns all items before the end index

negative number which returns all items before the end index from the end of the array or string

Examples:

{{ ['a','b','c','d']| slice:2}} will display c,d

{{ ['a','b','c','d']| slice:1:3}} will display b,c

{{ 'Laptop Charger'| slice:3:6}} will display top

{{ 'Laptop Charger'| slice:-4}} will display rger

{{ 'Laptop Charger'| slice:-4:-2}} will display rg

number

This pipe can be used to format a number.

Syntax:

{{ expression | number:digitInfo }}

The expression should be numeric

digitInfo is a string in the following format

{minIntegerDigits}.{minFractionDigits} - {maxFractionDigits}

minIntegerDigits is the minimum integer digits to display. The default value is 1.

minFractionDigits is the minimum number of digits to display after the fraction. The default value is 0.

maxFractionDigits is the maximum number of digits to display after fraction. The default value is 3.

Example:

{{ 25000 | number }} will display 25,000

{{ 25000 | number:'.3-5' }} will display 25,000.000

**Program:**

**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;

}

**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>

**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

],

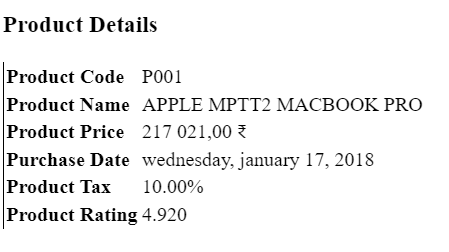
providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**output:**



**5c) 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. https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_24231999287700136000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

* Nested component is a component that is loaded into another component
* The component where another component is loaded onto is called a container component/parent component.
* The root component is loaded in the index.html page using its selector name. Similarly, to load one component into a parent component, use the selector name of the component in the template i.e., the HTML page of the container component
* Create a component called the 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

**EXERCISE-6**

6.aCourse 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 https://infyspringboard.onwingspan.com/web/en/viewer/webmodule/lex\_15758356947336235000\_shared?collectionId=lex\_208585155432546000 00\_shared&collectionType=Course

**Program:**

<div>

<h1>Welcome to Course App</h1>

<label for="courseSelect">Select a Course:</label>

<select id="courseSelect" [(ngModel)]="selectedCourse">

<option \*ngFor="let course of courses" [value]="course">History 01</option>

<option \*ngFor="let course of courses" [value]="course">Science 02</option>

<option \*ngFor="let course of courses" [value]="course">Maths 03</option>

<option \*ngFor="let course of courses" [value]="course">Biology 04</option>

<option \*ngFor="let course of courses" [value]="course">Economics</option>

</select>

<app-courses-list [selectedCourse]="selectedCourse"></app-courses-list>

</div>

<div \*ngIf="selectedCourse">

<h2>Course Details</h2>

<table>

<tr>

<th>Course Name</th>

<th>Instructor</th>

<th>Start Date</th>

</tr>

</table>

</div>

6b.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 th

**app.component.html**

<div>

<h1>Welcome to Course Registration App</h1>

<app-courses-list></app-courses-list>

</div>

**courses-list.component.html**

<table>

<thead>

<tr>

<th>Course Name</th>

<th>Instructor</th>

<th>Start Date</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let course of courses">

<td>{{ course.name }}</td>

<td>{{ course.instructor }}</td>

<td>{{ course.startDate }}</td>

<td><button (click)="register(course)">Register</button></td>

</tr>

</tbody>

</table>

**courses-list.component.t**s

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

@Component({

selector: 'app-courses-list',

templateUrl: './courses-list.component.html',

styleUrls: ['./courses-list.component.css']

})

export class CoursesListComponent {

courses = [

{ name: 'Math 101', instructor: 'John Doe', startDate: '2023-09-15' },

{ name: 'History 202', instructor: 'Jane Smith', startDate: '2023-09-20' },

{ name: 'Science 303', instructor: 'David Johnson', startDate: '2023-09-25' },

];

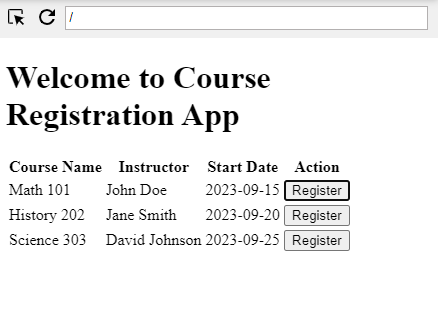
register(course: any) {

alert(`You have registered for ${course.name}`);

}

}

**Output**:



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

**ShadowDOM:**

**shadow-dom-example.component.ts**

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

@Component({

selector: 'app-shadow-dom-example',

templateUrl: './shadow-dom-example.component.html',

styleUrls: ['./shadow-dom-example.component.css'],

encapsulation: ViewEncapsulation.ShadowDom })

export class ShadowDomExampleComponent {

title = 'This content is inside the Shadow DOM';

}

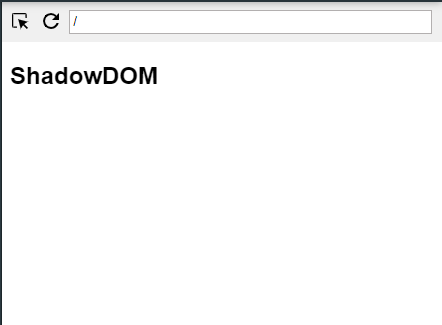
**shadow-dom-example.component.html**

<!-- src/app/shadow-dom-example/shadow-dom-example.component.html -->

<div class="shadow-dom-example">

<p>This content is inside the Shadow DOM.</p>

</div>



**None Encapsulation**

none-encapsulation-example.component.ts

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

@Component({

selector: 'app-none-encapsulation-example',

templateUrl: './none-encapsulation-example.component.html',

styleUrls: ['./none-encapsulation-example.component.css'],

encapsulation: ViewEncapsulation.None // Apply None encapsulation mode

})

export class NoneEncapsulationExampleComponent {

title = 'This content is inside the NoneEncapsulation';

}

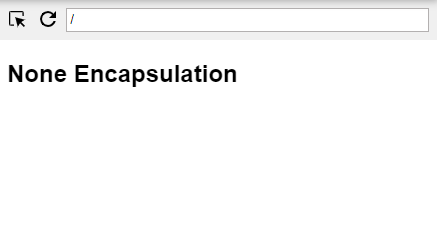
none-encapsulation-example.component.html

<!-- src/app/none-encapsulation-example/none-encapsulation-example.component.html -->

<div class="none-encapsulation-example">

<p>This content is not encapsulated and can affect other components.</p>

</div>



6d. Course Name: Angular JS Module Name: Component Life Cycle

Override component life-cycle hooks and logging the corresponding messages to understand the flow.

**app.component.html:-**

<p>hello world</p>

**app.component.ts:-**

import { Component, OnChanges, SimpleChanges } from '@angular/core';

@Component({

  selector: 'my-app',

  template: '<p>hello preethi</p>' // Add an empty template or a placeholder

})

export class ExampleComponent implements OnChanges {

  ngOnChanges(changes: SimpleChanges) {

    console.log('ngOnChanges:', changes);

  }

}

**app.module.ts:-**

import { NgModule } from '@angular/core';

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

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

@NgModule({

  imports: [

    BrowserModule

  ],

  declarations: [

    ExampleComponent

  ],

  bootstrap: [ExampleComponent]

})

export class AppModule { }

**Output:**

Helloworld

**EXERCISE-7**

**7 a.)Course Name: Angular JS Module Name: Template Driven Forms Create a courseregistration form as a template-driven form.**

**course-registration-form.component.htm**l

<form #courseRegistrationForm="ngForm" (ngSubmit)="submitForm(courseRegistrationForm)">

  <div class="form-group">

    <label for="name">Full Name:</label>

    <input type="text" id="name" name="name" ngModel required>

  </div>

  <div class="form-group">

    <label for="email">Email:</label>

    <input type="email" id="email" name="email" ngModel required email>

  </div>

  <div class="form-group">

    <label for="phone">Phone Number:</label>

    <input type="tel" id="phone" name="phone" ngModel pattern="[0-9]\*" minlength="10" maxlength="15" required>

  </div>

  <div class="form-group">

    <label for="course">Select a Course:</label>

    <select id="course" name="course" ngModel required>

      <option value="">Select a course</option>

      <option value="course1">Course 1</option>

      <option value="course2">Course 2</option>

      <option value="course3">Course 3</option>

    </select>

  </div>

  <div class="form-group">

    <label for="comments">Comments:</label>

    <textarea id="comments" name="comments" ngModel></textarea>

  </div>

 <button type="submit" [disabled]="!courseRegistrationForm.valid">Submit</button>

</form>

**course-registration-form.component.ts**

// Import necessary modules

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

@Component({

  selector: 'app-course-registration-form',

  templateUrl: './course-registration-form.component.html',

  styleUrls: ['./course-registration-form.component.css']

})

export class CourseRegistrationFormComponent {

  constructor() { }

  submitForm(form: any) {

    if (form.valid) {

      // You can access the form values through the 'form.value' object

      console.log('Form submitted with values:', form.value);

      // You can send the form data to your server or perform other actions here

    } else {

      // Handle form validation errors

      console.log('Form is invalid. Please check the fields.');

    }

  }

}

**app.component**.css

body {

  font-family: Arial, sans-serif;

  margin: 0;

  padding: 0;

  background-color: #f4f4f4;

}

.container {

  max-width: 800px;

  margin: 0 auto;

  padding: 20px;

  background-color: #fff;

  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

h1 {

  font-size: 24px;

  margin-bottom: 20px;

}

**app.component.html**

<div class="container">

  <h1>{{ title }}</h1>

  <app-course-registration-form></app-course-registration-form>

</div>

app.component.ts

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

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'Course Registration App';

}

**app.module.ts**

import { NgModule } from '@angular/core';

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

import { FormsModule } from '@angular/forms';

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

import { CourseRegistrationFormComponent } from './course-registration-form/course-registration-form.component';

@NgModule({

  declarations: [

    AppComponent,

    CourseRegistrationFormComponent

  ],

  imports: [

    BrowserModule,

    FormsModule

  ],

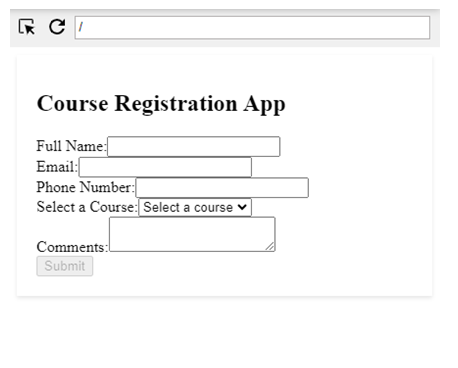
  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Output:**



7b.**Course Name:** Angular JS

**Module Name:** Model Driven Forms or Reactive Forms

Create an employee registration form as a reactive form.

**app.component.html:**-

<div>

  <h2>Employee Registration Form</h2>

  <form [formGroup]="employeeForm" (ngSubmit)="onSubmit()">

    <div class="form-group">

      <label for="firstName">First Name:</label>

      <input type="text" id="firstName" formControlName="firstName" class="form-control">

    </div>

    <div class="form-group">

      <label for="lastName">Last Name:</label>

      <input type="text" id="lastName" formControlName="lastName" class="form-control">

    </div>

    <div class="form-group">

      <label for="email">Email:</label>

      <input type="email" id="email" formControlName="email" class="form-control">

    </div>

    <div class="form-group">

      <label for="phone">Phone Number:</label>

      <input type="tel" id="phone" formControlName="phone" class="form-control">

    </div>

    <button type="submit" class="btn btn-primary">Submit</button>

  </form>

  <div class="message">

    {{ formMessage }}

  </div>

</div>

**app.component.ts:-**

import { FormBuilder, FormGroup } from '@angular/forms';

@Component({

 selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent implements OnInit {

  employeeForm: FormGroup;

  formMessage: string = '';

  constructor(private fb: FormBuilder) {}

  ngOnInit() {

    this.employeeForm = this.fb.group({

      firstName: [''],

      lastName: [''],

      email: [''],

      phone: ['']

    });

  }

  onSubmit() {

    if (this.employeeForm.valid) {

      console.log(this.employeeForm.value);

      this.formMessage = 'Data submitted successfully!';

      this.employeeForm.reset();

    } else {

      this.formMessage = 'Fill in all details correctly.';

    }

  }

}

**app.module.ts:-**

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

import { NgModule } from '@angular/core';

import { ReactiveFormsModule } from '@angular/forms'; // Import this line

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

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    ReactiveFormsModule, // Add this line

  ],

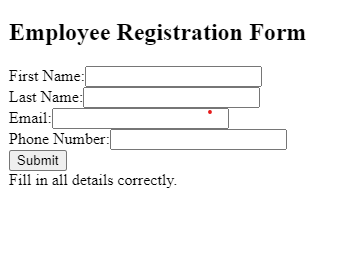
  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Output:**



7c.**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:**

**app.component.html:-**

<div>

  <h2>Employee Registration Form</h2>

  <form [formGroup]="employeeForm" (ngSubmit)="onSubmit()">

    <div class="form-group">

      <label for="firstName">First Name:</label>

      <input type="text" id="firstName" formControlName="firstName" class="form-control">

    </div>

    <div class="form-group">

      <label for="lastName">Last Name:</label>

      <input type="text" id="lastName" formControlName="lastName" class="form-control">

    </div>

    <div class="form-group">

      <label for="email">Email:</label>

      <input type="email" id="email" formControlName="email" class="form-control">

    </div>

    <div class="form-group">

      <label for="phone">Phone Number:</label>

      <input type="tel" id="phone" formControlName="phone" class="form-control">

    </div>

    <button type="submit" class="btn btn-primary">Submit</button>

  </form>

 <div class="message">

    {{ formMessage }}

  </div>

</div>

**app.component.ts:-**

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

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent implements OnInit {

  employeeForm: FormGroup;

  formMessage: string = '';

  constructor(private fb: FormBuilder) {}

  ngOnInit() {

    this.employeeForm = this.fb.group({

      firstName: ['', Validators.required],

      lastName: ['', Validators.required],

      email: ['', [Validators.required, Validators.email]],

      phone: ['', [Validators.required, Validators.pattern(/^\d{10}$/)]]

    });

  }

  onSubmit() {

    if (this.employeeForm.valid) {

      this.formMessage = 'Data submitted successfully!';

      this.employeeForm.reset();

    } else {

      this.formMessage = 'Fill in all details correctly.';

    }

  }

}

**app.module.ts:-**

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

import { NgModule } from '@angular/core';

import { ReactiveFormsModule } from '@angular/forms'; // Import this line

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

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    ReactiveFormsModule, // Add this line

  ],

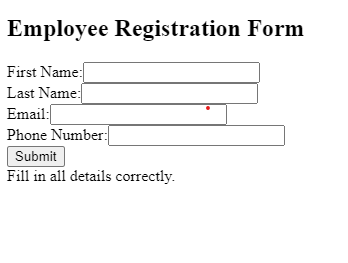
  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Output:**



**EXERCISE-8**

8A.**Course Name:** Angular JS

**Module Name:** Custom Validators in Template Driven forms

Create a custom validator for the email field in the course registration form.

**app.component.html:-**

<form [formGroup]="registrationForm">

  <!-- Other form fields -->

  <div class="form-group">

    <label for="email">Email:</label>

    <input type="email" id="email" class="form-control" formControlName="email">

    <!-- Display a message when email is valid -->

    <div \*ngIf="submitted && registrationForm.get('email').valid" class="text-success">

      Correct

    </div>

    <!-- Display an error message when email is invalid -->

    <div \*ngIf="submitted && registrationForm.get('email').hasError('invalidEmail')" class="text-danger">

      Invalid email format.

    </div>

  </div>

  <!-- Submit button -->

  <button (click)="onSubmit()">Submit</button>

</form>

**app.component.ts:-**

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

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { emailValidator } from './app.module'; // Import the custom validator

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

})

export class AppComponent {

  registrationForm: FormGroup;

  submitted = false; // Add a flag to track form submission

  constructor(private fb: FormBuilder) {

    this.registrationForm = this.fb.group({

      email: ['', [Validators.required, emailValidator]], // Use the custom validator

    });

  }

  onSubmit() {

    this.submitted = true; // Set the flag to true on submit

    if (this.registrationForm.valid) {

      console.log('Form submitted successfully');

    }

  }

}

app.module.ts:-

import { NgModule } from '@angular/core';

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

import { FormsModule } from '@angular/forms';

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

import { ReactiveFormsModule } from '@angular/forms';

@NgModule({

  declarations: [AppComponent],

  imports: [BrowserModule, ReactiveFormsModule],

  bootstrap: [AppComponent],

})

export class AppModule {}

export function emailValidator(control: any) {

  const emailPattern = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;

  if (control.value && !emailPattern.test(control.value)) {

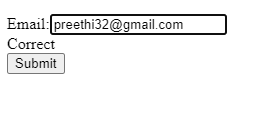
    return { invalidEmail: true };

  }

  return null;

}

**Output:**



8b.**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:**

**app.component.html:-**

<!-- app.component.html -->

<div>

  <h1>Welcome to my-app</h1>

  <app-book-list></app-book-list>

</div>

**app.component.ts:-**

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

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  name = 'Angular ' + VERSION.full;

}

**app.module.ts:-**

import { NgModule } from '@angular/core';

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

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

import { BookListComponent } from './book-list/book-list.component';

@NgModule({

  imports: [

    BrowserModule

  ],

  declarations: [

    AppComponent,

    BookListComponent

  ],

  bootstrap: [AppComponent]

})

export class AppModule { }

book-list.component.html:-

<!-- book-list.component.html -->

<div>

  <h2>Book List</h2>

  <ul>

    <li \*ngFor="let book of books">

      {{ book.name }}

    </li>

  </ul>

</div>

**book-list.component.ts:-**

// book-list.component.ts

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

import { BookService } from './book.service';

@Component({

  selector: 'app-book-list',

  templateUrl: './book-list.component.html',

  styleUrls: ['./book-list.component.css'],

})

export class BookListComponent implements OnInit {

  books: { id: number; name: string }[] = [];

 constructor(private bookService: BookService) {}

  ngOnInit(): void {

    this.bookService.getBooks().subscribe((books) => {

      this.books = books;

    });

  }

}

**book.service.ts:-**

**// book.service.ts**

import { Injectable } from '@angular/core';

import { Observable, of } from 'rxjs';

@Injectable({

  providedIn: 'root',

})

export class BookService {

  private books: { id: number; name: string }[] = [

    { id: 1, name: 'Book 1' },

    { id: 2, name: 'Book 2' },

    { id: 3, name: 'Book 3' },

  ];

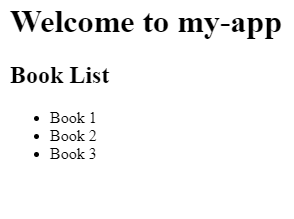
  getBooks(): Observable<{ id: number; name: string }[]> {

    return of(this.books);

  }

}

**Output:**



8c.**Course Name:** Angular JS

**Module Name:** RxJS Observables

Create and use an observable in Angular.

**Program:**

**app.module.ts:-**

import { NgModule } from '@angular/core';

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

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

@NgModule({

  imports: [

    BrowserModule

  ],

  declarations: [

    AppComponent

  ],

  bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.ts:-**

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

import { DataService } from './data.service';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

})

export class AppComponent implements OnInit {

  data: string;

constructor(private dataService: DataService) {}

  ngOnInit(): void {

    this.dataService.getData().subscribe((result) => {

      this.data = result;

    });

  }

}

**app.component.html:-**

<div>

  <h1>Welcome to my-app</h1>

  <p>Data from Observable:</p>

  <p>{{ data }}</p>

</div>

data.service.ts:-

import { Injectable } from '@angular/core';

import { Observable, of } from 'rxjs';

@Injectable({

  providedIn: 'root',

})

export class DataService {

  getData(): Observable<string> {

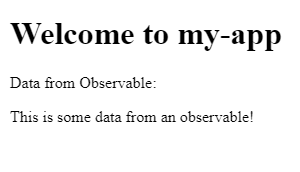
    const data = 'This is some data from an observable!';

    return of(data);

  }

}

**Output:**



**EXERCISE-9**

**Course Name:** Angular JS

**Module Name:** Server Communication using HttpClient Create an application for Server Communication using HttpClient

**Step 1: Create a Service**

Name:data

Step 2: Implement the Service

Data.service.ts

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class DataService {

private apiUrl = 'https://jsonplaceholder.typicode.com'; // Replace with your server URL

constructor(private http: HttpClient) { }

getData(): Observable<any> {

return this.http.get(`${this.apiUrl}/posts/1`); // Replace with your API endpoint

}

}

Step 4: Create a Component

Name:data-display

data-display.component.ts

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

import { DataService } from '../data.service';

@Component({

selector: 'app-data-display',

templateUrl: './data-display.component.html',

styleUrls: ['./data-display.component.css']

})

export class DataDisplayComponent implements OnInit {

data: any;

constructor(private dataService: DataService) { }

ngOnInit(): void {

this.dataService.getData().subscribe(response => {

this.data = response;

});

}

}

Step 5: Create a Component Template

data-display.component.html

<div \*ngIf="data">

<h2>Data from the Server:</h2>

<pre>{{ data | json }}</pre>

</div>

app.component.html:

<app-data-display></app-data-display>

app.module.ts:

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

import { NgModule } from '@angular/core';

import { HttpClientModule } from '@angular/common/http'; // Import HttpClientModule

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

import { DataDisplayComponent } from './data-display/data-display.component';

import { DataService } from './data.service';

@NgModule({

declarations: [

AppComponent,

DataDisplayComponent

],

imports: [

BrowserModule,

HttpClientModule, // Add HttpClientModule to the imports array

],

providers: [DataService],

bootstrap: [AppComponent]

})

export class AppModule { }

Output:

Data from the Server:

{

"userId": 1,

"id": 1,

"title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

"body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

}

**Output:**

The data received from the server in JSON format. It includes the following fields:

userId : The ID of the user.

Id : The ID of the data entry.

Title : The title of the data entry.

Body : The body or content of the data entry.

This is a typical response from a JSON API, and your Angular application has successfully fetched and displayed this data from the server. You can now use this data as needed within your application.

**9b. Course Name:** Angular JS

**Module Name:** Communicating with different backend services using Angular HttpClient Create a custom service called ProductService in which Http class is used to fetch data stored in the JSON files.

**Product-list.component.ts**:

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

import { ProductService } from './product.service';

@Component({

  selector: 'app-product-list',

  templateUrl: './product-list.component.html',

  styleUrls: ['./product-list.component.css']

})

export class ProductListComponent implements OnInit {

  products: any[] = [];

  constructor(private productService: ProductService) { }

  ngOnInit() {

    this.productService.getProducts().subscribe((data) => {

      this.products = data;

    });

  }

}

**Product.service.ts:**

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class ProductService {

  private productsUrl = 'assets/products.json';

  constructor(private http: HttpClient) { }

  getProducts(): Observable<any[]> {

    return this.http.get<any[]>(this.productsUrl);

  }

}

**App.component.html:**

<div \*ngIf="products.length > 0; else noProducts">

  <h2>Products</h2>

  <ul>

    <li \*ngFor="let product of products">

      <h3>{{ product.name }}</h3>

      <p>Price: {{ product.price }}</p>

      <p>Description: {{ product.description }}</p>

    </li>

  </ul>

</div>

<ng-template #noProducts>

  <p>No products available</p>

</ng-template>

**App.component.ts:**

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

import { ProductService } from './product-list/product.service';

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent implements OnInit {

  products: any[] = [];

  constructor(private productService: ProductService) { }

  ngOnInit() {

    this.productService.getProducts().subscribe((data) => {

      this.products = data;

    });

  }

}

**App.module.ts**:

import { NgModule } from '@angular/core';

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

import { HttpClientModule } from '@angular/common/http';

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

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    HttpClientModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Exercise - 10**

**Course Name**: Angular JS

**Module Name**: Routing Basics, Router Links Create multiple components and add routing to provide navigation between them.

**app-routing.module.ts:**

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { HomeComponent } from './home/home.component';

import { AboutComponent } from './about/about.component';

const routes: Routes = [

  { path: 'home', component: HomeComponent },

  { path: 'about', component: AboutComponent },

  { path: '', redirectTo: '/home', pathMatch: 'full' },

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**app.component.html:**

<nav>

  <a routerLink="/home">Home</a>

  <a routerLink="/about">About</a>

</nav>

<router-outlet> </router-outlet>

**app.component.ts:**

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

@Component({

  selector: 'my-app',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

}

**app.module.ts:**

import { NgModule } from '@angular/core';

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

import { RouterModule } from '@angular/router'; // Import the RouterModule

import { AppRoutingModule } from './app-routing.module';

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

@NgModule({

  declarations: [AppComponent],

  imports: [BrowserModule, AppRoutingModule, RouterModule], // Add RouterModule here

  bootstrap: [AppComponent]

})

export class AppModule { }

**about.component.html:**

<p>

  mst

</p>

**home.component.html:**

<p>

  Mean stack technologies

</p>

**Output:**

HomeAbout

**Mst**

HomeAbout

**Mean stack technologies**

# 10)b)Module Name:Route Guards

# Considering the same example used for routing, add route guard to BooksComponent. Only after logging in, the user should be able to access BooksComponent. If the user tries to give the URL of Bookscomponent in another tab or window, or if the user triesConsidering the same example used for routing, add route guard to BooksComponent. Only after logging in, the user should be able to access BooksComponent. If the user tries to give the URL of Bookscomponent in another tab or window, or if the user tries

import React from 'react';

import { BrowserRouter as Router, Route, Link } from 'react-router-dom'; import Home from './components/Home';

import About from './components/About'; import Contact from './components/Contact'; const App = () => {

return (

<Router>

<div>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/about">About</Link>

</li>

<li>

<Link to="/contact">Contact</Link>

</li>

</ul>

</nav>

<hr />

<Route exact path="/" component={Home} />

<Route path="/about" component={About} />

<Route path="/contact" component={Contact} />

</div>

</Router>

);

};

export default App; import React from 'react'; const Contact = () => {

return <div>This is the Contact page.</div>;

};

export default Contact;

# 10)c)Module Name:Asynchronous Routing

**Apply lazy loading to BookComponent. If lazy loading is not added to the demo, it has loaded in 1.14 s. Observe the load time at the bottom of the browser console. Press F12 in the browser and click the Network tab and check the Load time**

# Code:

import React from 'react'; const Books = () => {

return <div>This is the Books page. Only logged-in users can access this.</div>;

};

export default Books;

import React, { Suspense } from 'react';

import { BrowserRouter as Router, Route, Link } from 'react-router-dom'; import Home from './components/Home';

import About from './components/About'; import Contact from './components/Contact';

import { login, logout, isAuthenticatedUser } from './authService'; const Books = React.lazy(() => import('./components/Books')); const App = () => {

// ... rest of the code ... return (

<Router>

<div>

<hr />

<Suspense fallback={<div>Loading...</div>}>

<Route path="/books" component={Books} />

</Suspense>

</div>

</Router>

);

};

export default App;

# 10)d)Module Name:Nested Routes Implement Child Routes to a submodule Code:

// src/components/BookDetail.js import React from 'react';

import { useParams } from 'react-router-dom'; const BookDetail = () => {

const { bookId } = useParams();

return <div>Details for Book {bookId}</div>;

};

export default BookDetail;

// src/components/Books.js import React from 'react';

import { Route, Link, useRouteMatch } from 'react-router-dom'; import BookDetail from './BookDetail';

const Books = () => {

const { path, url } = useRouteMatch(); return (

<div>

<h2>Books</h2>

<ul>

<li>

<Link to={`${url}/1`}>Book 1</Link>

</li>

<li>

<Link to={`${url}/2`}>Book 2</Link>

</li>

{/\* Add more books as needed \*/}

</ul>

<Route exact path={path}>

<p>Please select a book.</p>

</Route>

<Route path={`${path}/:bookId`} component={BookDetail} />

</div>

);

};

export default Books;

// src/App.js

import React, { Suspense } from 'react';

import { BrowserRouter as Router, Route, Link } from 'react-router-dom'; import Home from './components/Home';

import About from './components/About'; import Contact from './components/Contact';

import { login, logout, isAuthenticatedUser } from './authService'; const Books = React.lazy(() => import('./components/Books')); const App = () => {

// ... rest of the code ... return (

<Router>

<div>

{/\* ... navigation and login/logout buttons ... \*/}

<hr />

<Suspense fallback={<div>Loading...</div>}>

<Route path="/books" component={Books} />

</Suspense>

</div>

</Router>

);

};

export default App;

**Exercise – 11**

# 11 a. MongoDB Essentials - A Complete MongoDB Guide Module Name: Installing MongoDB on the local computer, Create MongoDB Atlas Cluster

**Installing MongoDB on Local Computer:**

* + - **Download MongoDB:** Visit the official MongoDB website (<https://www.mongodb.com/try/download/community>) to download the MongoDB Community Server for your operating system.
    - **Install MongoDB:**
      * Windows: Run the downloaded installer and follow the installation wizard's instructions.
      * macOS: Use Homebrew to install MongoDB by running **brew tap mongodb/brew** followed by **brew install mongodb/brew/mongodb-community**.
      * Linux: Follow the instructions provided on the MongoDB website for your specific distribution.
    - **Start MongoDB:**
      * Windows: MongoDB is typically installed as a service and starts automatically. You can also manually start it using the Services application.
      * macOS and Linux: Start MongoDB using the terminal by running
* **mongod**.
  + - **Access MongoDB Shell:**
      * Open a new terminal window.
      * Run **mongo** to start the MongoDB shell and interact with the database.
* **Creating a MongoDB Atlas Cluster:**
* MongoDB Atlas is a cloud-based MongoDB service that provides fully managed databases.
* **Sign Up for MongoDB Atlas:**
  + Go to the MongoDB Atlas website (<https://www.mongodb.com/cloud/atlas>) and click "Get started free."
  + Follow the prompts to sign up for an account.

**1. Create a Cluster:**

* After signing up and logging in, click the "Build a New Cluster" button.
  + Choose a cloud provider, region, and cluster settings. You can choose the free tier (M0) to get started.

1. **Configure Security:**
   * Click "Security" in the left-hand menu.
   * Add your IP address to the IP Whitelist to allow connections to your cluster.
2. **Connect to Your Cluster:**
   * Click "Clusters" in the left-hand menu.
   * Click the "Connect" button for your cluster.
   * Choose "Connect Your Application" to get the connection string.
3. **Connect to Atlas Cluster using MongoDB Shell:**
   * Open a terminal window.
   * Run **mongo "your\_connection\_string"** to connect to your Atlas cluster. Replace **"your\_connection\_string"** with the actual connection string you obtained.

11.b. **Introduction to the CRUD Operations Write MongoDB queries to perform CRUD operations on document using insert(), find(), update(), remove().**

**SOURCE CODE:**

1. **Create (Insert) Documents**:

To insert documents into a collection, you can use the **insertOne()** or **insertMany()**

methods.

// Insert a single document

db.collectionName.insertOne({ field1: value1, field2: value2 })

// Insert multiple documents db.collectionName.insertMany([

{ field1: value1, field2: value2 },

{ field1: value3, field2: value4 }

])

1. **Read (Find) Documents**:

To retrieve documents from a collection, you can use the **find()** method.

// Find all documents in a collection db.collectionName.find()

// Find documents with a specific field value db.collectionName.find({ field: value })

// Find documents with a field value matching a range db.collectionName.find({ age: { $gt: 25, $lt: 40 } })

1. **Update Documents**:

To update documents in a collection, you can use the **updateOne()** or

**updateMany()** methods. db.collectionName.updateOne(

{ field: value },

{ $set: { updatedField: newValue } }

)

// Update multiple documents db.collectionName.updateMany(

{ field: value },

{ $set: { updatedField: newValue } }

)

1. **Delete (Remove) Documents**:

To remove documents from a collection, you can use the **deleteOne()** or

**deleteMany()** methods. db.collectionName.deleteOne({ field: value }) db.collectionName.deleteMany({ field: value })

**Exercise - 12**

* 1. **12a. Create and Delete Databases and Collections Write MongoDB queries to Create and drop databases and collections.**
  2. To create a new database, you can use the **use** command in the MongoDB shell. However, note that a database isn't actually created until you insert data into it.

// Switch to a new database (creates it if it doesn't exist) use newDatabaseName

* + 1. **Create a Collection**:

Collections are created automatically when you insert data into them. However, you can explicitly create a collection using the **createCollection()** method.

// Create a new collection in the current database db.createCollection("newCollectionName")

* + 1. **Drop a Collection**:

To drop (delete) a collection, you can use the **drop()** method.

// Drop a collection db.collectionName.drop()

* + 1. **Drop a Database**:

To drop a database, you can use the **dropDatabase()** method. Make sure to switch to the appropriate database before using this command.

// Drop the current database (make sure you're in the right database) db.dropDatabase().

* 1. **12b. Create and Delete Databases and Collections**

Write MongoDB queries to Create and drop databases and collections. SOURCE CODE:

**find()**: The **find()** method retrieves documents from a collection that match a specified query. You can use various query operators to filter the results.

// Find all documents in a collection db.collectionName.find()

// Find documents with a specific field value db.collectionName.find({ field: value })

// Find documents with a field value matching a range db.collectionName.find({ age: { $gt: 25, $lt: 40 } })\

**limit()**: The **limit()** method restricts the number of documents returned by a query.

// Limit the number of documents returned db.collectionName.find().limit(10)

**sort()**: The **sort()** method arranges the documents in a specific order based on one or more fields.

// Sort documents in ascending order by a field db.collectionName.find().sort({ field: 1 })

// Sort documents in descending order by a field db.collectionName.find().sort({ field: -1 })

**createIndex()**: The **createIndex()** method creates an index on specified fields in a collection, which can improve query performance.

// Create an ascending index on a field db.collectionName.createIndex({ field: 1 })

// Create a descending index on a field db.collectionName.createIndex({ field: -1 })

**aggregate()**: The **aggregate()** method performs advanced data processing using a pipeline of stages. It's useful for complex querying and transformations.

// Example aggregation pipeline: calculate average age by gender db.collectionName.aggregate([

{ $group: { \_id: "$gender", avgAge: { $avg: "$age" } } }