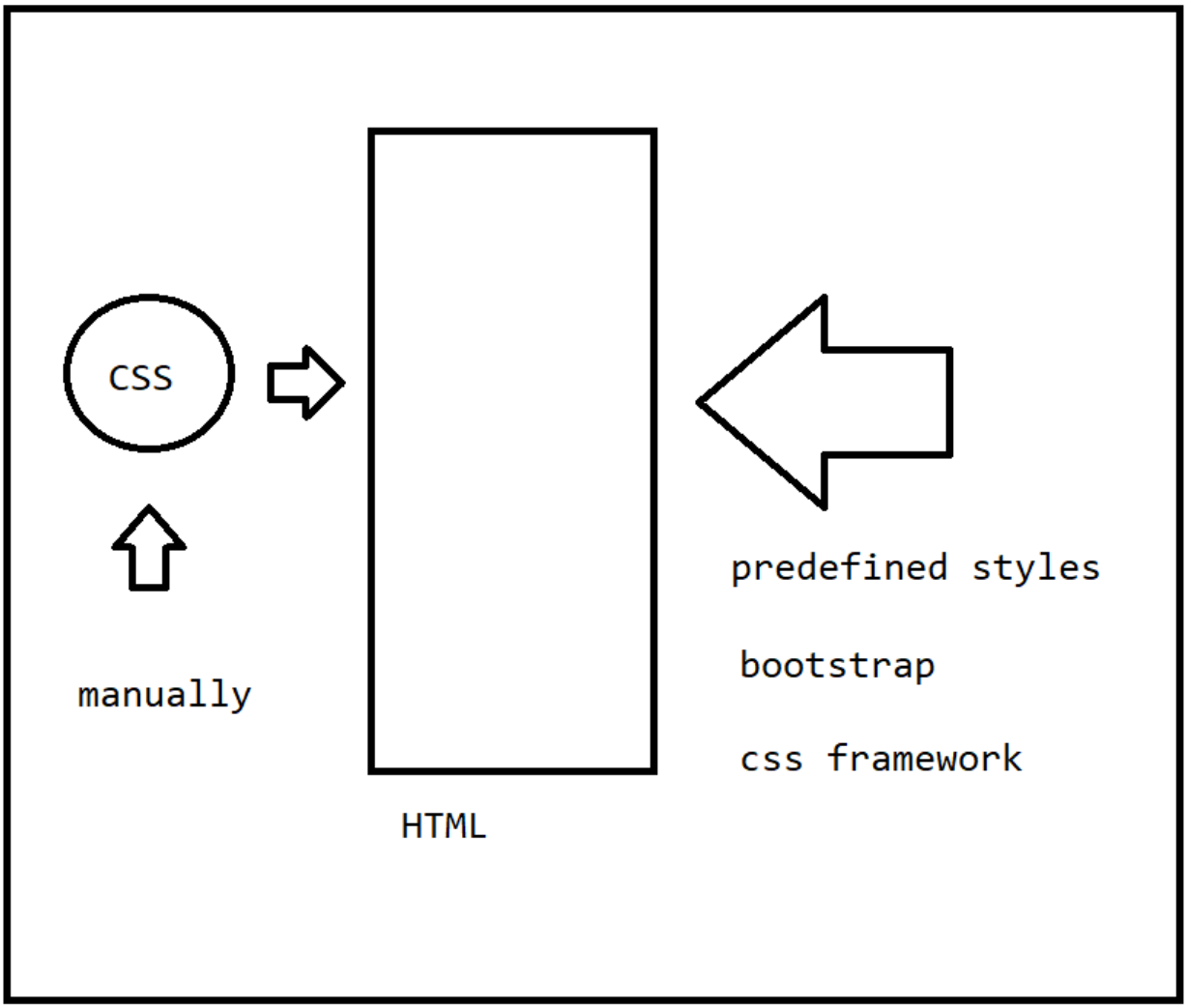
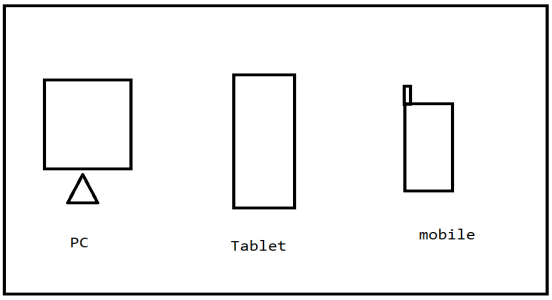
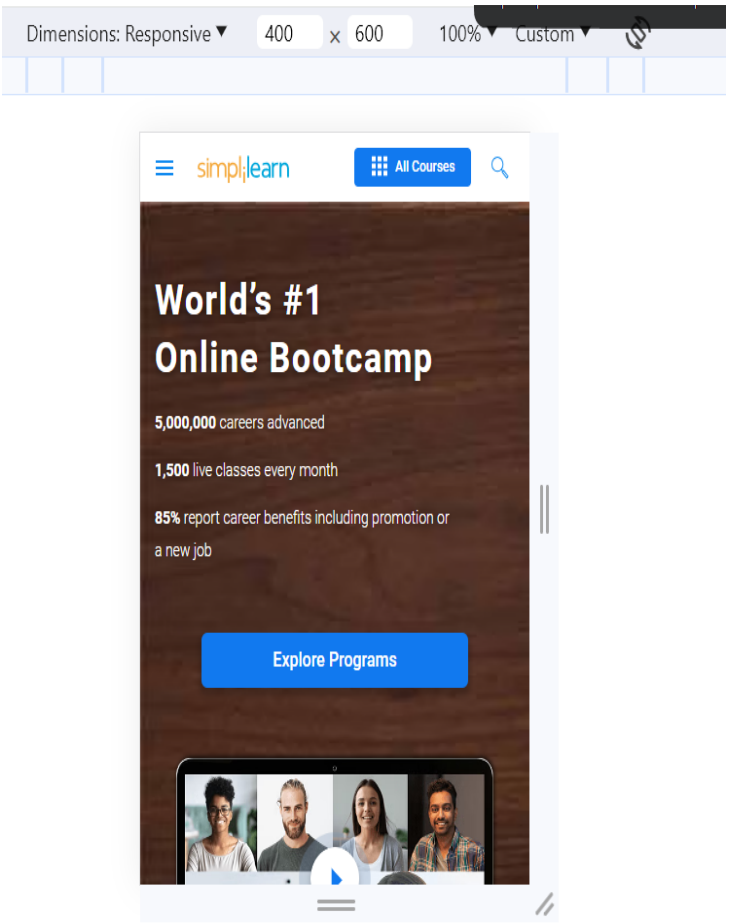
Bootstrap :



It is a framework of css and an open-source
It provides a ready-made structure of the CSS components for the tags
Response oriented screens:



=> cntrl+shift+i , cntrl+shft+m

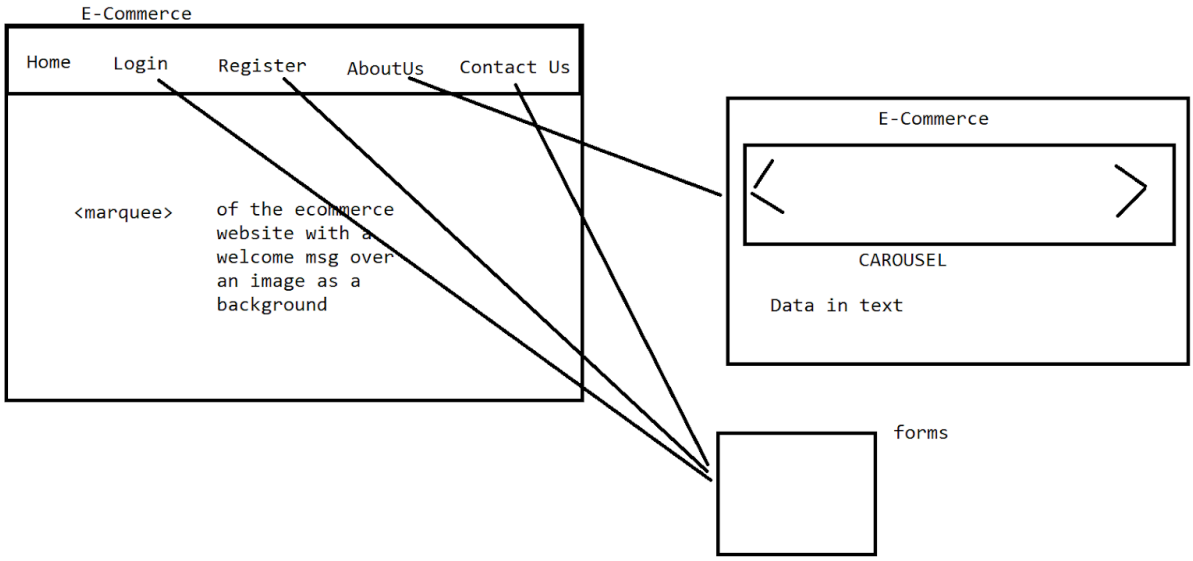


https://getbootstrap.com/docs/5.0/content/typography/

Cheetsheet of 5.0

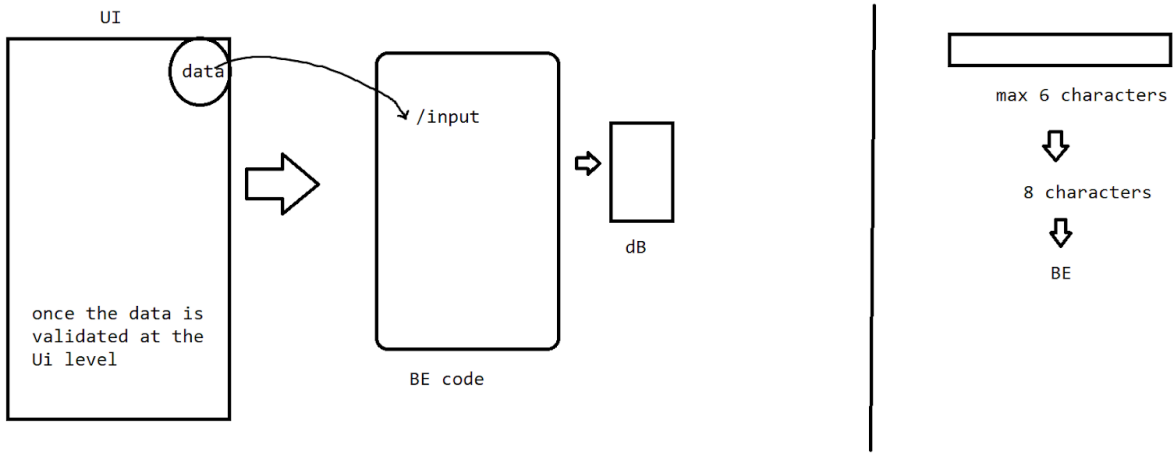
https://bootstrap-cheatsheet.themeselection.com/

POC : Any theme



Include a Cart with an image + cost =>buy button ⇒pagination

=>Javascript



It is a scripting-based language - HTML components - > Object based rules

HTML =>static pages =>(not processing data)
HTML + js =>dynamic page => response

It is a light weighted language

- It is used to validate the data over the client side only, by reducing the burden of the server side.
- Js is going to be loaded by the browser itself
- Js as a dynamically typed language [int a, float b , String ….]
  var,let, const
- Js uses the functional approach

Limitation: It doesn't have any idea about the things that happen at the BE.

JS invoke -> an action =>event =>event handlers at HTML side

<head> section =>define the js

We click on a button =>action generated => function validation(){}

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
    function msg(){
    document.write("hi learners welcome to javascript session  !!!")
    }
    </script>
</head>
<body>
   <h1> Javascript lession 1</h1>
   <form>
    <input type="button" value="click" onclick="msg()" >
   </form>
</body>
</html>
```

- Note: Need to write the HTML and javascript separately to make the code look more loosely coupled.

External js

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <script src="config.js">
    </script>
</head>
<body>
   <h1> Javascript lession 1</h1>
   <form>
    <input type="button" value="click" onclick="msg()" >
   </form>
</body>
</html>
```

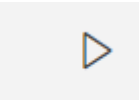
Config.js

```
function msg(){
    document.write("hi learners welcome to javascript session  !!!")
    }
```

Task :

If my javascript is present on any drive then how i can have a location in the script-src???

1. In the workspace create a folder . how u can access?
2. In D:/ create a folder how can u access??

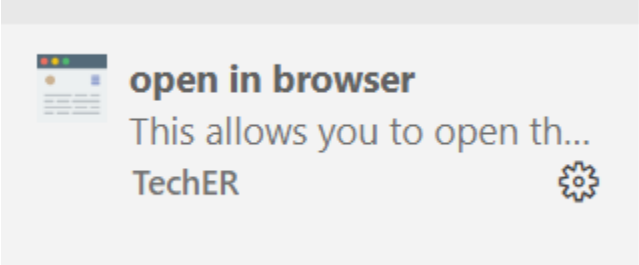


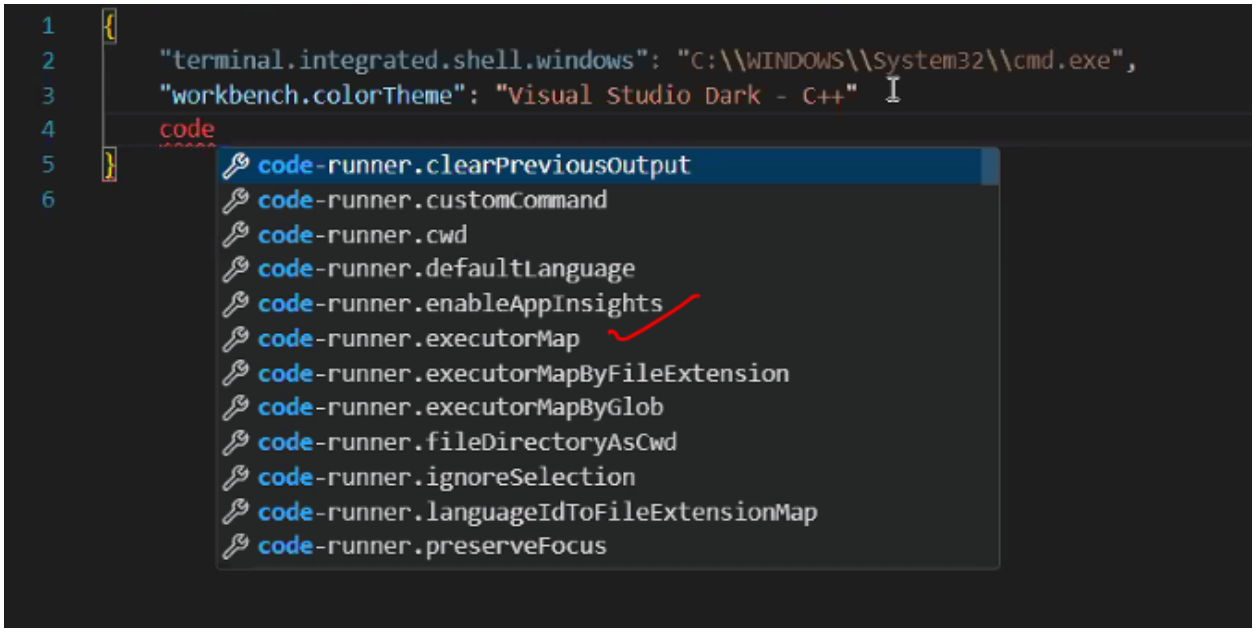
If the server is not working effectively use the run button

C: > Users > Karthik > AppData > Roaming > Code > User > {} settings.json > ...

```
{
    "code-runner.executorMap": {
        "html": "start chrome",
        "javascript": "node",
```

Open in browser as extension






```
{
    "code-runner.executorMap": {
        "html": "start chrome",
        "javascript": "node",
        "java": "cd $dir && javac $fileName && java $fileNameWithoutExt",
        "c": "cd $dir && gcc $fileName -o $fileNameWithoutExt &&
$dir$fileNameWithoutExt",
        "cpp": "cd $dir && g++ $fileName -o $fileNameWithoutExt &&
$dir$fileNameWithoutExt",
        "objective-c": "cd $dir && gcc -framework Cocoa $fileName -o
$fileNameWithoutExt && $dir$fileNameWithoutExt",
        "php": "php",
        "python": "python -u",
        "perl": "perl",
        "perl6": "perl6",
        "ruby": "ruby",
        "go": "go run",
        "lua": "lua",
        "groovy": "groovy",
        "powershell": "powershell -ExecutionPolicy ByPass -File",
        "bat": "cmd /c",
        "shellscript": "bash",
        "fsharp": "fsi",
        "csharp": "scriptcs",
        "vbscript": "cscript //Nologo",
```

```
        "typescript": "ts-node",
        "coffeescript": "coffee",
        "scala": "scala",
        "swift": "swift",
        "julia": "julia",
        "crystal": "crystal",
        "ocaml": "ocaml",
        "r": "Rscript",
        "applescript": "osascript",
        "clojure": "lein exec",
        "haxe": "haxe --cwd $dirWithoutTrailingSlash --run
$fileNameWithoutExt",
        "rust": "cd $dir && rustc $fileName && $dir$fileNameWithoutExt",
        "racket": "racket",
        "scheme": "csi -script",
        "ahk": "autohotkey",
        "autoit": "autoit3",
        "dart": "dart",
        "pascal": "cd $dir && fpc $fileName && $dir$fileNameWithoutExt",
        "d": "cd $dir && dmd $fileName && $dir$fileNameWithoutExt",
        "haskell": "runhaskell",
        "nim": "nim compile --verbosity:0 --hints:off --run",
        "lisp": "sbcl --script",
        "kit": "kitc --run",
        "v": "v run",
        "sass": "sass --style expanded",
        "scss": "scss --style expanded",
        "less": "cd $dir && lessc $fileName $fileNameWithoutExt.css",
        "FortranFreeForm": "cd $dir && gfortran $fileName -o
$fileNameWithoutExt && $dir$fileNameWithoutExt",
        "fortran-modern": "cd $dir && gfortran $fileName -o
$fileNameWithoutExt && $dir$fileNameWithoutExt",
        "fortran_fixed-form": "cd $dir && gfortran $fileName -o
$fileNameWithoutExt && $dir$fileNameWithoutExt",
        "fortran": "cd $dir && gfortran $fileName -o $fileNameWithoutExt
&& $dir$fileNameWithoutExt",
        "sml": "cd $dir && sml $fileName"
    },
    "window.zoomLevel": 2,
    "workbench.colorTheme": "Visual Studio Light",
```

```
    "javascript.updateImportsOnFileMove.enabled": "always"
}
```

//dynamically typed language

```
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
   var s="the output is "
   function msg(){
    var a=10
    var b=5.3
    document.write(s+(a+b)/2)
    }
    </script>
</head>
<body>
   <h1> Javascript lession 1</h1>
   <form>
    <input type="button" value="click" onclick="msg()" >
   </form>
</body>
</html>
```

In js no need to mention a datatype , the datatype is going to get auto assigned upon the data what you pass .

//Object based topics - refer lession 11 -Self learning

HTML => 3 components of pop up boxes

1. Alert
2. Confirm
3. Prompt

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
   <script>
   var s="the output is "
   function msg(){
    var a=10
    var b=5.3
    alert(s+(a+b)/2)
    }
    </script>
</head>
<body>
   <h1> Javascript lession 1</h1>
   <form>
    <input type="button" value="click" onclick="msg()" >
   </form>
</body>
</html>
```

Confirm :
When we have 2 cases which one case needs to get chosen ?

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
   var s="the output is "
   function msg(){
   var input=window.confirm("do you want to proceed??")
   if(input==true){
      document.write("going to main logic ")
   }
   else{
      alert("the application gets terminated ..")
   }
    }
    </script>
</head>
```

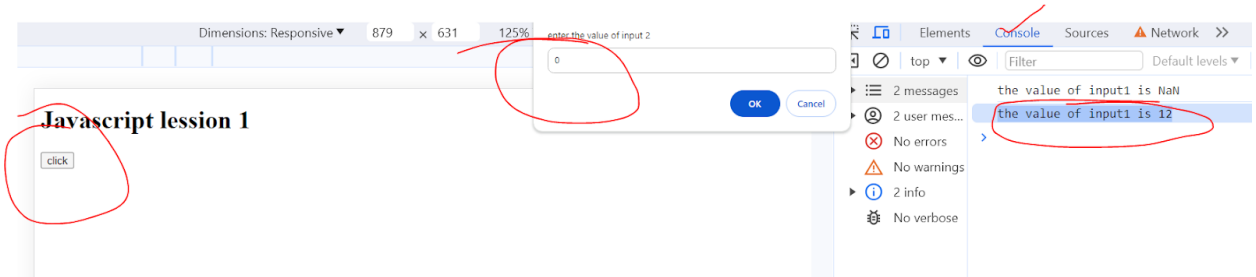
```
<body>
   <h1> Javascript lession 1</h1>
   <form>
    <input type="button" value="click" onclick="msg()" >
   </form>
</body>


</html>
```

#prompt

```
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
   var s="the output is "
   function msg(){
  var input1=parseInt(window.prompt("enter the value of input 1 ","0"))
  var input2=parseInt(window.prompt("enter the value of input 2 ","0"))
  alert("the out put of this 2 numbers is :"+(input1+input2))
    }
    </script>
</head>
<body>
   <h1> Javascript lession 1</h1>
   <form>
    <input type="button" value="click" onclick="msg()" >
   </form>
</body>
</html>
```

- We use loggers as the debuggers of the values

```
console.log("the value of input1 is "+input1)
```



Task 👍
Take a number in the prompt and validate it as a prime or not , if not a prime raise an alert .

//validate a form –client validation

```
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
  function checkvalidationOfForm(){
   let name=document.f1.user.value;
   console.log("the name is entered is "+name)
   let pass=document.f1.pwd.value;
   console.log("the pwd is entered is "+pass)

   if(name==""){
      alert("name should not be empty ")
   }
   else if(pass==""){
      alert("password must not be empty")
   }
   else if(name=="admin" && pass=="admin"){
      document.write("Hey Welcome "+name)
   }
   else{
      alert("please check credentials ")
   }

  }
    </script>
```

```
</head>
<body>
   <h1> Login Form</h1>
   <form name="f1">
      username<input type="text" name="user"><br>
      password<input type="password" name="pwd"><br>
      <input type="button" value="login" onclick="checkvalidationOfForm()"
>

   </form>
</body>
</html>
```

//isue with multiple alerts

```
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
  function checkvalidationOfForm(){
   let name=document.f1.user.value;
   console.log("the name is entered is "+name)
   let pass=document.f1.pwd.value;
   console.log("the pwd is entered is "+pass)

   if(name==""){
      alert("name should not be empty ")
   }
  if(pass==""){
      alert("password must not be empty")
   }
  if(name.length<6){
      alert("name must not be <6")
   }
   if(name==="admin" && pass==="admin"){
      document.write("Hey Welcome "+name)
   }
   else{
      alert("please check credentials ")
   }
```

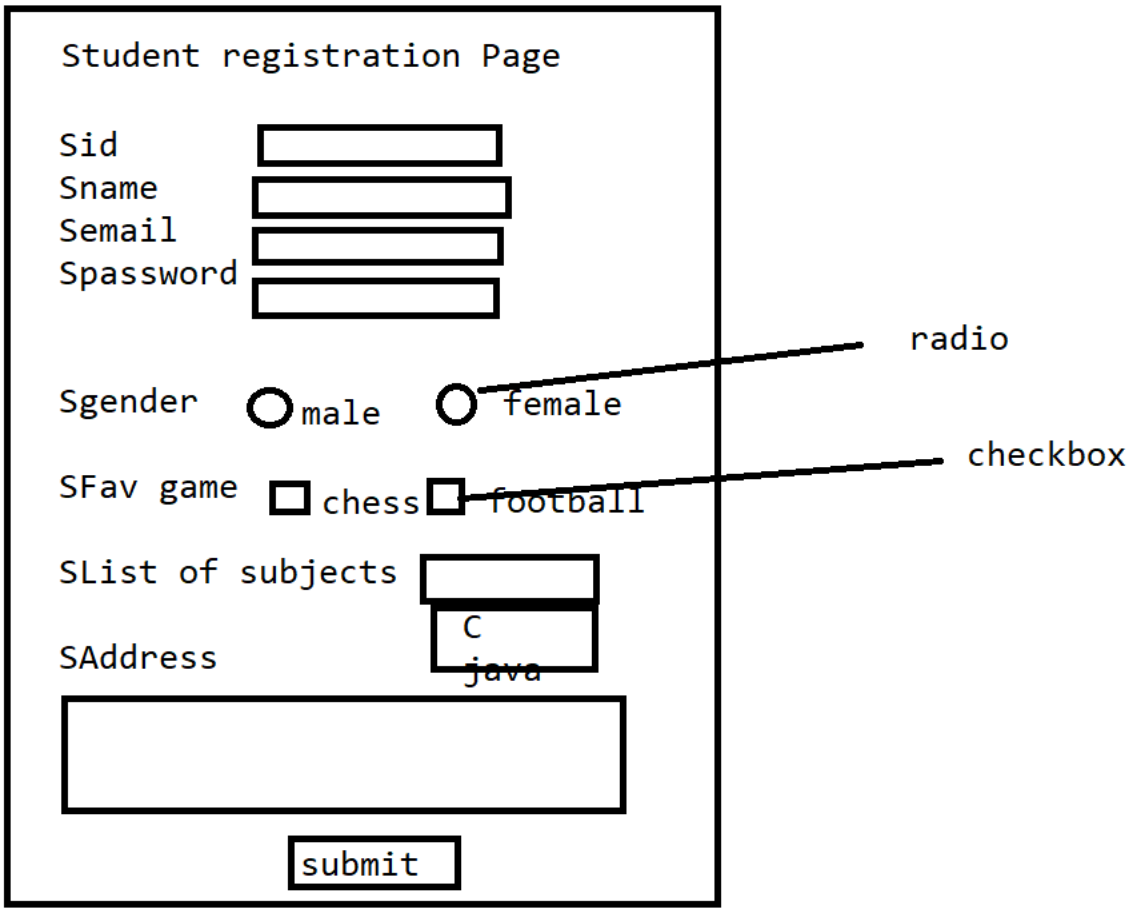
```
  }
    </script>
</head>
<body>
   <h1> Login Form</h1>
   <form name="f1">
      username<input type="text" name="user"><br>
      password<input type="password" name="pwd"><br>
      <input type="button" value="login" onclick="checkvalidationOfForm()"
>

   </form>
</body>
</html>
```

//solution

```
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
  function checkvalidationOfForm(){
   let name=document.f1.user.value;
   console.log("the name is entered is "+name)
   let pass=document.f1.pwd.value;
   console.log("the pwd is entered is "+pass)
   let nameStatusofEmpty=""
let passStatusofEmpty=""
let namelengthStatus=""
let other=""
   if(name==""){
      nameStatusofEmpty="name should not be empty "
   }
  if(pass==""){
   passStatusofEmpty="password must not be empty"
   }
  if(name.length<6){
   namelengthStatus="name must not be <6"
   }
```

```
   if(name===="admin" && pass===="admin"){
      document.write("Hey Welcome "+name)
   }
   else{
      other="please check credentials "
   }

alert(nameStatusofEmpty+"\n"+passStatusofEmpty+"\n"+namelengthStatus+"\n"+
other)

  }
    </script>
</head>
<body>
   <h1> Login Form</h1>
   <form name="f1">
      username<input type="text" name="user"><br>
      password<input type="password" name="pwd"><br>
      <input type="button" value="login" onclick="checkvalidationOfForm()"
>

   </form>
</body>
</html>
```

Task : 10 min



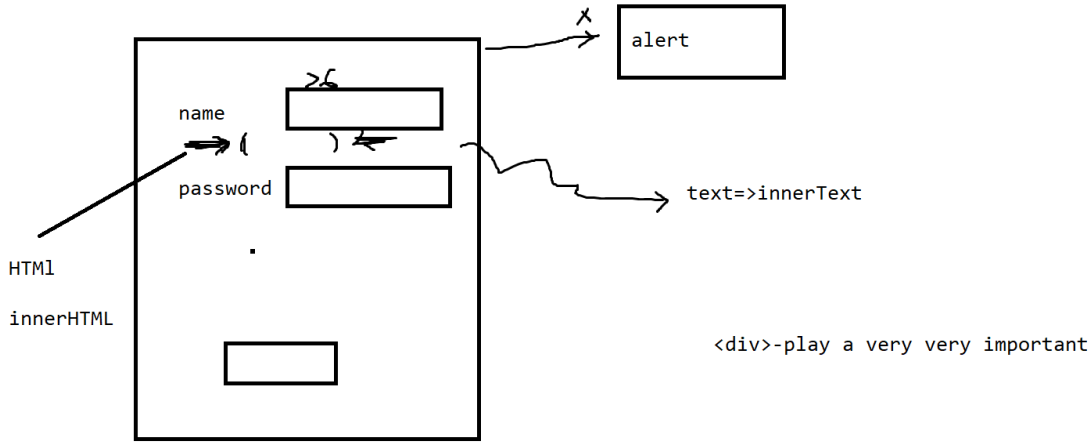
Validate the data with the corresponding alerts by checking these fields are not empty

If registration => a statement -> registration is successfully validated …!

- Upgrade this by using arrays push
  //let status=[]
  status.push(“errormsg”)
  If length>0
  alert()

=> get the value of radio button and check box into the console

InnerHTML and innerText




```
<!DOCTYPE html>
<html lang="en">
<head>
   <script src="config.js">

    </script>
</head>
<body>
   <input type="button" value="click for login form"
onclick="generateLoginForm()">
<div id="place"></div>

</body>
</html>


function generateLoginForm(){
    var tag="<form name='f1'> username<input type='text'
name='user'><br>password<input type='password'
name='pwd'><br></form>"
    document.getElementById("place").innerHTML=tag
```

```
}
```

- Note : Any HTML component , make a practice of written it in a

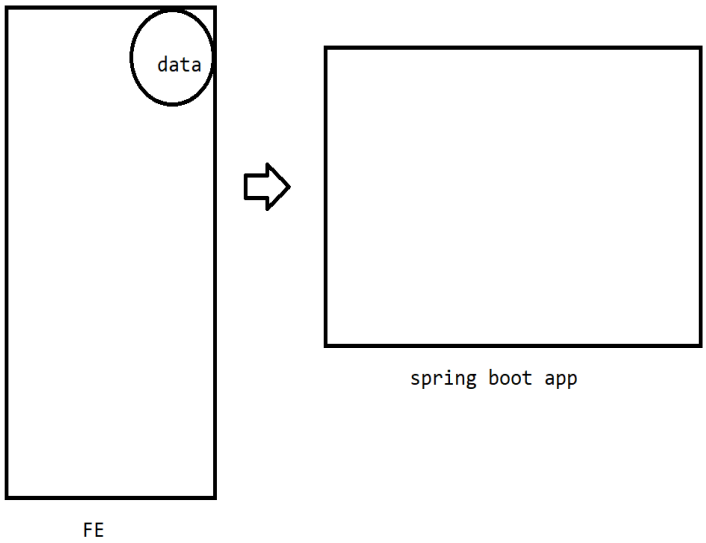
//innerText

```
<!DOCTYPE html>
<html lang="en">
<head>
   <script>
  function checkvalidationOfForm(){
   let name=document.f1.user.value;
   console.log("the name is entered is "+name)
   let pass=document.f1.pwd.value;
   console.log("the pwd is entered is "+pass)
   let nameStatusofEmpty=""
let passStatusofEmpty=""
let namelengthStatus=""
let other=""
   if(name==""){
      nameStatusofEmpty="name should not be empty "
      document.getElementById("user").innerText=nameStatusofEmpty
   }
  if(pass==""){
   passStatusofEmpty="password must not be empty"
   document.getElementById("pass").innerText=passStatusofEmpty
   }
   if(name=="admin" && pass=="admin"){
      document.write("Hey Welcome "+name)
   }



  }
    </script>
</head>
<body>
   <h1> Login Form</h1>
   <form name="f1">
      username<input type="text" name="user"><br>
      <div id="user" style="color: chocolate;"></div>
```

```
      password<input type="password" name="pwd" ><br>
      <div id="pass" style="color: chocolate;"></div>
      <input type="button" value="login"
onclick="checkvalidationOfForm()" >

   </form>
</body>
</html>
```

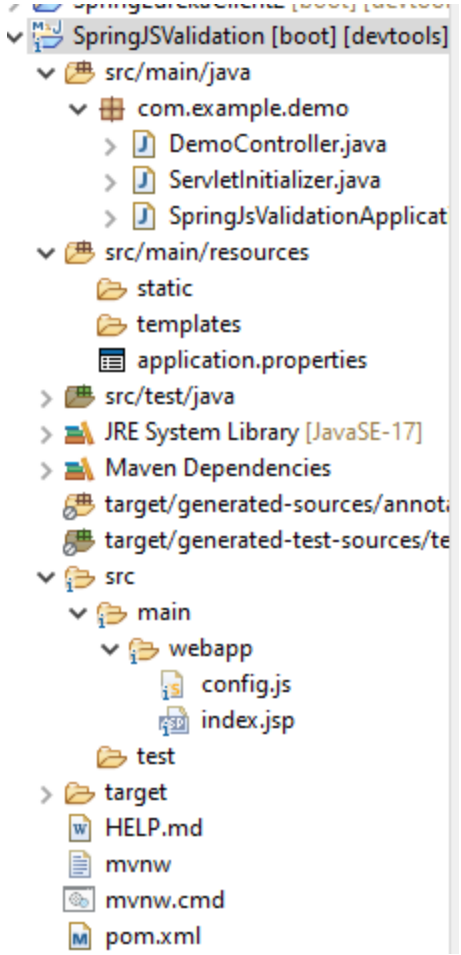
Task :

Upgrade registration validation to innerText



1. Create a webpage
2. Go to the function and validate the data
3. Use the XMLHttpRequest to push the data onto the spring boot app.

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jasper</artifactId>
  <version>9.0.82</version>
</dependency>
```



Index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
<script type="text/javascript" src="config.js"></script>
</head>
<body>
<form name="f1">
Email<input type="text" name="email"><br>
<div id="place1" style="color: red;"></div>
Phono<input type="text" name="phono"><br>
<div id="place2" style="color: red;"></div>
<input type="button" value="register" onclick="checkform()">
```

```
</form>
</body>
</html>
```

Config.js

```
function checkform(){
	var email=document.f1.email.value;
	var phono=document.f1.phono.value;
	var statusofemail=""
	var statusofphone=""
	if(email==""){
		statusofemail="email cannot be empty"
		document.getElementById("place1").innerText=statusofemail
	}

	if(phono==""){
		statusofphone="phono cannot be empty "
		document.getElementById("place2").innerText=statusofphone
	}

	if(statusofemail=="" && statusofphone==""){
	console.log("inside the rasing of request")
		sendRequest(email,phono);
	}
}



function sendRequest(email,phono){

	/*
	1.open the request
	2. set the header
	3. make it to ready state
	4. validate the ready sttate status
	5. send the request
	*/
	console.log("inside the request")

	var xhr=new XMLHttpRequest();

	//requesttype,reqname,enabler of the config
	xhr.open("POST","/demo",true)
```

```
	xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded")
	xhr.onreadystatechange=function(){
		if(xhr.readyState==XMLHttpRequest.DONE&&xhr.status==200){

		}
	};

	xhr.send("email="+email+"&phono="+phono)
}
```

```
package com.example.demo;

import java.util.logging.Logger;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;



@Controller
public class DemoController {

	Logger log=Logger.getAnonymousLogger();

	@RequestMapping("/demo")
	public void response(HttpServletRequest request,HttpServletResponse response) {
		log.info(request.getParameter("email")  +"    "+request.getParameter("phono"));
	}



}
```

// create an insert of data into the dB with the validation and display it - Home assignment

Registration Page

//json file → HTML via the javascript

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="quizconfig.js"></script>
</head>
<body>
    <h1>Quiz</h1>

    <div id="quiz-container">
            <h2 id="question"></h2>
            <ul id="choices"></ul>
            <button id="submit">submit</button>
    </div>

    <div id="result-container">
            <h2 id="result"></h2>

    </div>


</body>
</html>
```

//question.json

```json
{
"questions":[

{
    "question":"what is the capital of france?",
    "choices":["London","Paris","Rome","Berlin"],
    "correctAnswer":"Paris"
},
{

    "question":"India has provided a $250 million Line of Credit to which country for the modernization of its agricultural sector??",
    "choices":["Nigeria","South Africa","Ethiopia","Kenya"],
```

```
    "correctAnswer":"Kenya"

}
]
}
```

//quizconfig.js

```
fetch('question.json')
.then(response=>response.json())
.then(data=>{

const quizContainer=document.getElementById("quiz-container");
const quizElement=document.getElementById("question");
const choicesElement=document.getElementById("choices");
const submitButton=document.getElementById("submit");
const resultContainer=document.getElementById("result-container");
const resultElement=document.getElementById("result");


let currentQuestionIndex=0;
let score=0;


function loadQuestion(){

const currentQuestion=data.questions[currentQuestionIndex];
quizElement.textContent=currentQuestion.question;

choicesElement.innerHTML='';

currentQuestion.choices.forEach(choice => {

    const li=document.createElement('li');
    const input=document.createElement('input');
    input.setAttribute('type','radio');
    input.setAttribute('name','answer');
    input.setAttribute('value',choice);
```

```
    li.appendChild(input);
    li.appendChild(document.createTextNode(choice));
    choicesElement.appendChild(li);

});



}



function checkAnswers(){
    const
selectedAnswer=document.querySelector('input[name="answer"]:checked');
    if(selectedAnswer){
        console.log("selectedAnswer"+selectedAnswer)
        const userAnswer=selectedAnswer.value;
        const currentQuestion=data.questions[currentQuestionIndex];
        if(userAnswer===currentQuestion.correctAnswer){
            score++
        }
        currentQuestionIndex++;
        if(currentQuestionIndex<data.questions.length){
            loadQuestion();
        }
        else{
            showResult();
        }
    }
}

function showResult(){
    quizContainer.style.display='none';
    resultContainer.style.display='block';
    resultElement.textContent=`you scored ${score} out of
${data.questions.length} questions`;
}

submitButton.addEventListener('click',checkAnswers);
```

```
loadQuestion();

})
```

Task :

Make a json of 10 questions and perform the quiz

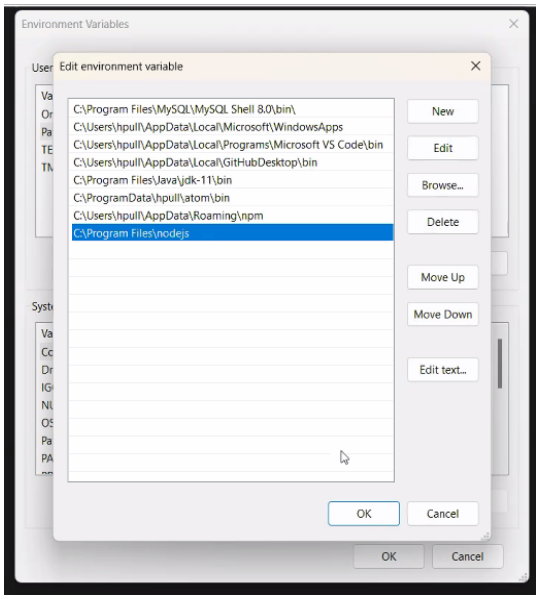
//Object based

Devasish - prototyping
Kamal - IIFE

Angular :

Prerequisite:
https://nodejs.org/en/download/current

Install node js and set the path

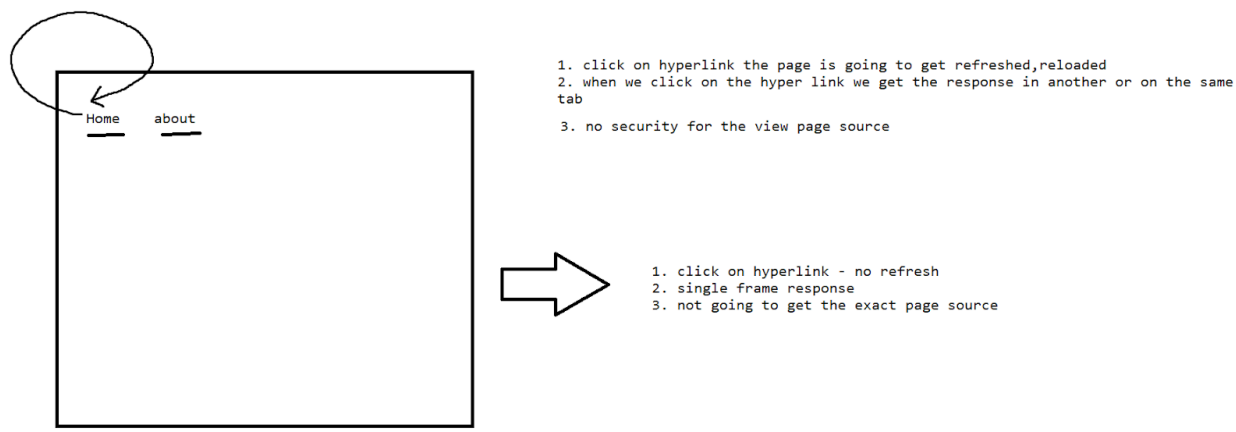



```
C:\Users\hpull>node -v
v20.10.0

C:\Users\hpull>
```

npm install -g @angular/cli

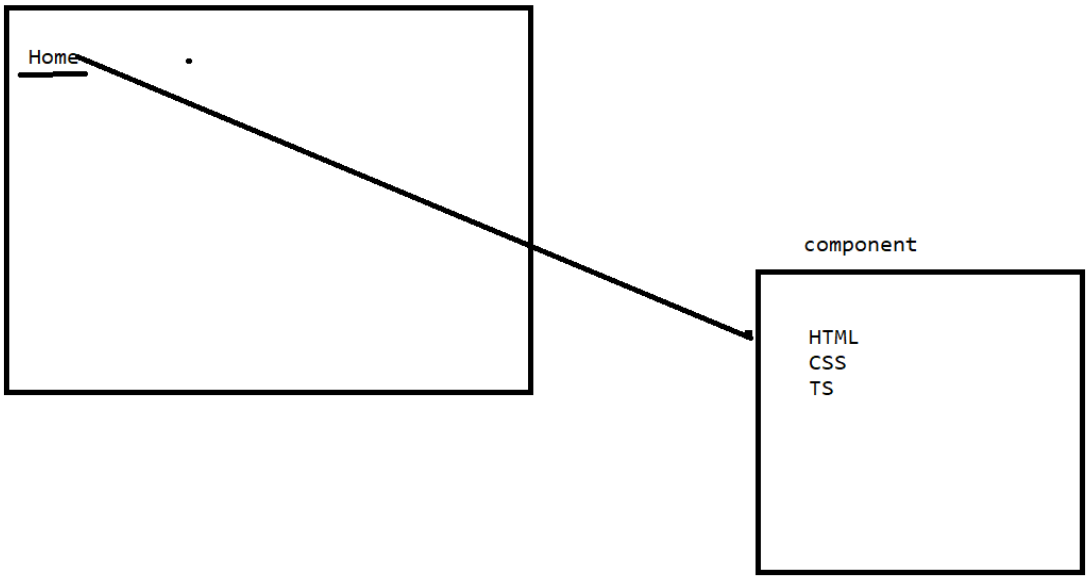
```
C:\Users\hpull>ng v
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable --global

Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled

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

Angular CLI: 17.0.7
Node: 20.10.0
Package Manager: npm 10.2.3
OS: win32 x64

Angular:
...
```

Angular :



Angular -> page ->component

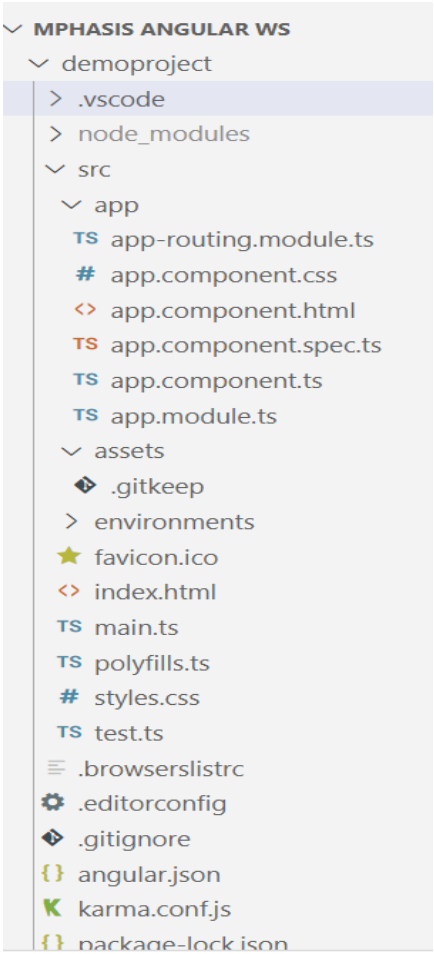
Component => HTML + CSS + TS(typescript)



How to create an angular project?

```
F:\mphasis angular ws>ng new demoproject
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE demoproject/angular.json (2947 bytes)
CREATE demoproject/package.json (1042 bytes)
CREATE demoproject/README.md (1065 bytes)
CREATE demoproject/tsconfig.json (863 bytes)
CREATE demoproject/.editorconfig (274 bytes)
CREATE demoproject/.gitignore (548 bytes)
CREATE demoproject/.browserslistrc (600 bytes)
```

```
REATE demoproject/src/app/app.component.css (0 bytes)
 Packages installed successfully.
arning: in the working copy of '.browserslistrc', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of '.editorconfig', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of '.vscode/extensions.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of '.vscode/launch.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of '.vscode/tasks.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'angular.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'karma.conf.js', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/app/app-routing.module.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/app/app.component.html', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/app/app.component.spec.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/app/app.component.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/app/app.module.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/environments/environment.prod.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/environments/environment.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/index.html', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/main.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/polyfills.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/styles.css', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'src/test.ts', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'tsconfig.app.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'tsconfig.json', LF will be replaced by CRLF the next time Git touches it
arning: in the working copy of 'tsconfig.spec.json', LF will be replaced by CRLF the next time Git touches it
   Successfully initialized git.
```



Note : Node: 21 and angular 17

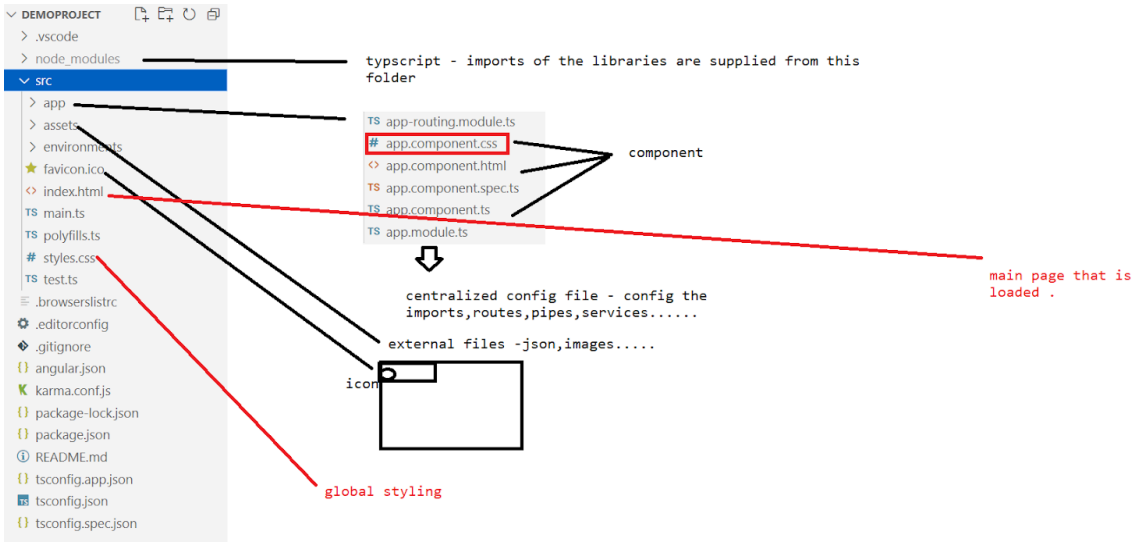
Create a project with the command

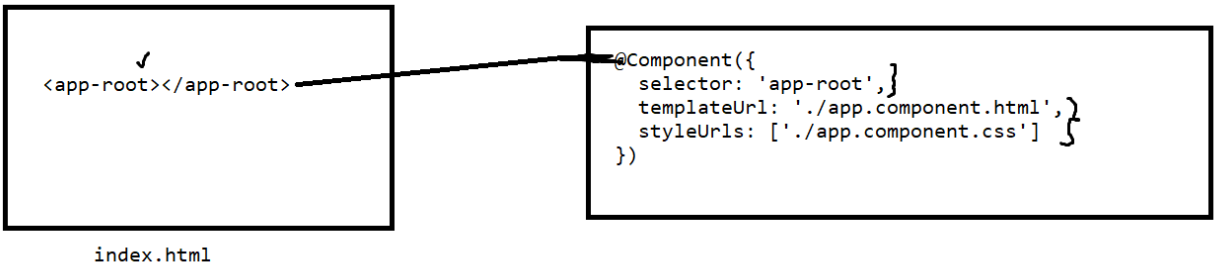
```
ng new --no-standalone <projectname>
```

Run angular project

```
F:\mphasis angular ws>cd demoproject

F:\mphasis angular ws\demoproject>ng serve -o
? 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. (y/N)
```

Angular default port is 4200





How to create a own component ??

```
√ Compiled successfully.
^C
F:\mphasis angular ws\demoproject>ng g c first
CREATE src/app/first/first.component.html (20 bytes)
CREATE src/app/first/first.component.spec.ts (592 bytes)
CREATE src/app/first/first.component.ts (271 bytes)
CREATE src/app/first/first.component.css (0 bytes)
UPDATE src/app/app.module.ts (471 bytes)

F:\mphasis angular ws\demoproject>
```

app.component.html M × | TS first.component.ts U

src > app > app.component.html > app-first

```
1  <h1>This is angular demo </h1>
2  <app-first></app-first>
```

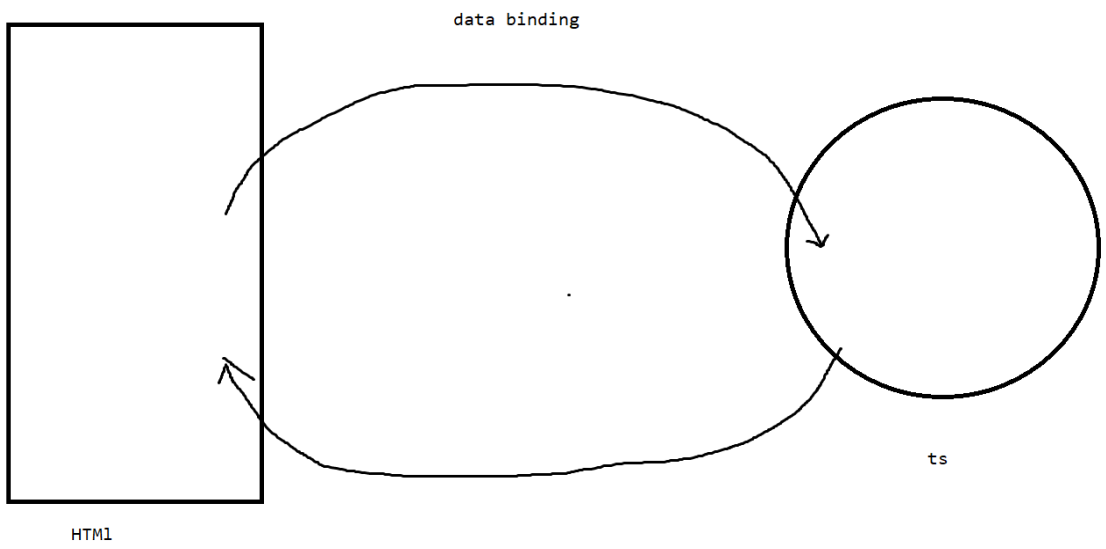
Task :

Create 3 page components a load them on a single app component page .

TypeScript: Javascript framework

- Dynamically typed language

>databinding



>one way binding
>two way binding

One way - recieving the data from the ts on to the HTML
Two way - recieving the data from the ts on to the HTML and vice versa

One way :
> String interpolation
>property binding
>class binding
>style binding
>event binding

String interpolation - ts -> {{}}
.ts

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

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

```
  title = 'demoproject';

  heading="session on angular"
}
```

.html

```
<h1>{{heading}}</h1>
<app-first></app-first>
```

//property binding

The below is the static way of defining the attributes

```
sid<input type="text" hidden>
<button disabled>submit</button>
```

Dynamic way - from the ts we operate the attribute values

[property]= value in the ts

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

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

  ishidden=false
  isdisabled=true

}
```

```
sid<input type="text" [hidden]="ishidden">
```

```
<button [disabled]="isdisabled">submit</button>
```

>class binding

[class]=”actionname”

```
<h1 [class]="isactive?'active':'inactive'"> Hi leaners welcome </h1>
```

//.css

```
.active{
    background-color: aquamarine;
    color: white;
}


.inactive{
    background-color: bisque;
    color: blue;
}
```

//.ts

```
export class AppComponent {
  title = 'demoproject';

  isactive=false

}
```

//style binding

[style]=”styleobjname”

Limitations: anything of the style ‘-’ will not be taken

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  active:object={color:'white',background:'grey'}

}
```

```
<h1 [style]="active"> Hi leaners welcome </h1>
```

Event binding :

(eventname)=”functionname”

```
<button (click)="incr()">increment</button>
<button (click)="decr()">decrement</button>

<h1>{{count}}</h1>
```

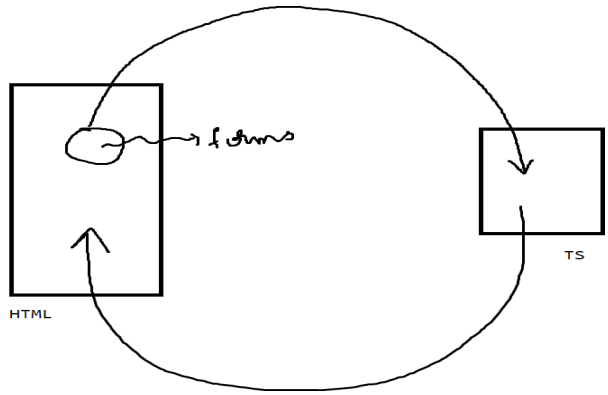
```
import { Component } from '@angular/core';

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

```
 incr(){
  this.count+=1;
 }

 decr(){
  this.count-=1
 }
}
```

Two way binding :

We need to add configuration FormsModule in “app.module.ts ”

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FirstComponent } from './first/first.component';
import { FormsModule } from '@angular/forms';

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

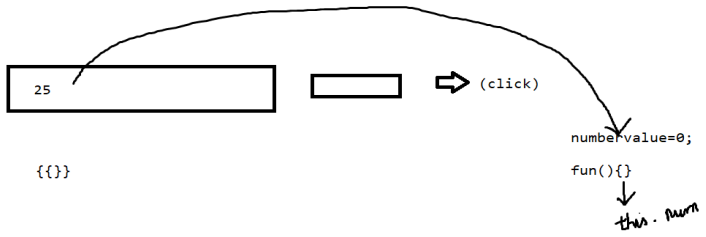



```
Enter a name :<input type="text" name="name" [(ngModel)]="tname"><br>

the entered name is {{tname}}
```

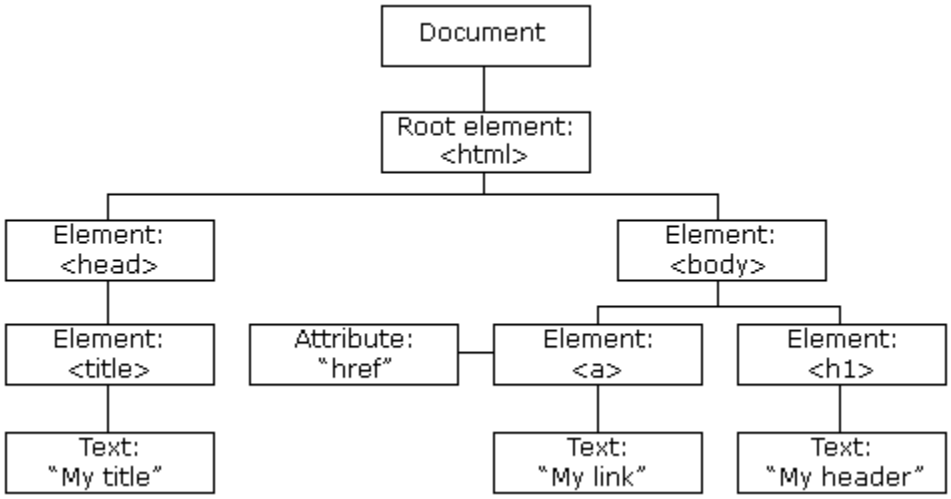
```
import { Component } from '@angular/core';

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

Task:

Take a field input of a number and check whether it is a palindrome or not by taking dynamic input.



DOM
Document object model :



Directives :

Custom HTML , attributes that tell the angular to change the structure , style, behaviour of the DOM

>Structural directives:
Change the DOM layout by adding/removing the DOM elements

ngIf,ngFor,ngSwitch

>ngIf
>ngIf="value"=>true
>ngIf with else

>ngIf with then else<stmt>

Model -1

```
<h1 *ngIf="show()">Student works!!</h1>

show(){
  return true
}
```

Model -2

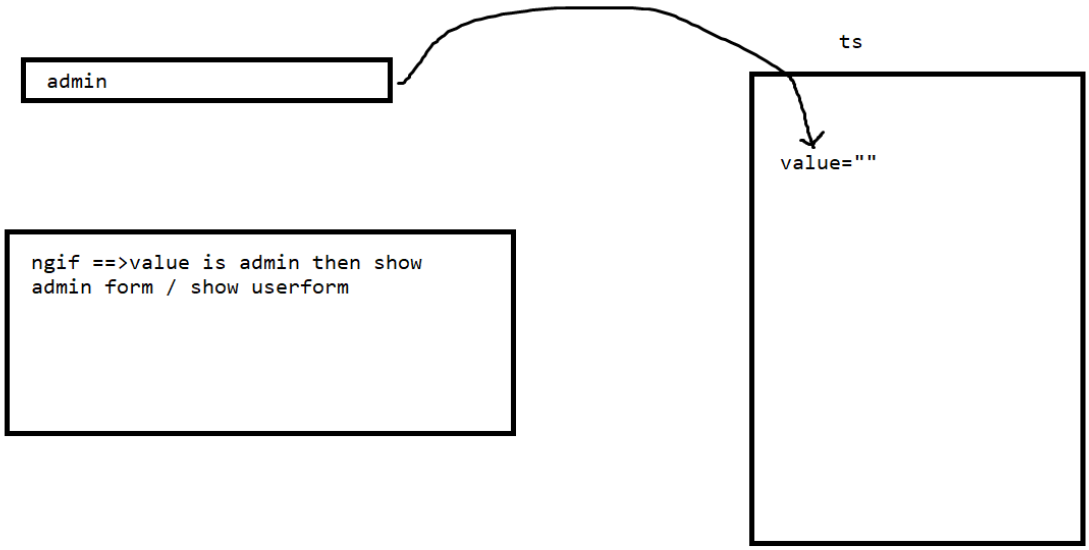
```
<h1 *ngIf="show else fail">Student works!!</h1>
<ng-template #fail>Name<input type="text"></ng-template>


show=true;
```

Model -3

```
<h1 *ngIf="show;then success;else fail"></h1>
<ng-template #success>Student works!!</ng-template>
<ng-template #fail>Name<input type="text"></ng-template>

show=true;
```

Task:

On a condition I need to get an admin login form/user login form




```
ngSwitch,ngSwitchCase,ngSwitchDefault
<div [ngSwitch]="opr">
<div *ngSwitchCase="'+'">{{num1+num2}}</div>
<div *ngSwitchCase="'-'">{{num1-num2}}</div>
<div *ngSwitchCase="'*'">{{num1*num2}}</div>
<div *ngSwitchCase="'/'">{{num1/num2}}</div>
<div *ngSwitchDefault>wrong option given</div>
</div>
```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
num1=2
num2=3
opr="*"
```

```
}
```

Task 👍

num1

num2

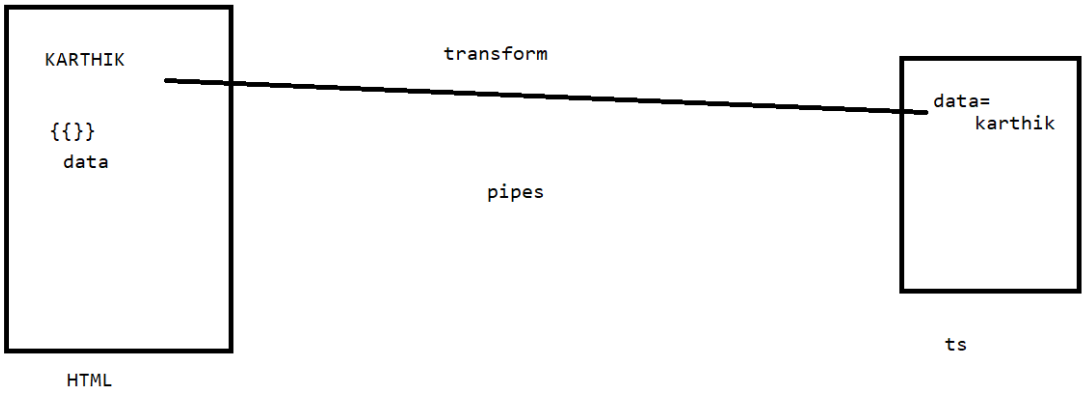
opr

The output is :

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
mobilebrand=[
{"id":1,"name":"karthik","mobile":"samsung"},
{"id":2,"name":"Veviek","mobile":"Iphone"}

]

}
```

```
<table border="1">
<tr><th>Id</th><th>Name</th><th>Mobile</th></tr>
<tr *ngFor="let m of mobilebrand">
<td>{{m.id}}</td>
<td>{{m.name}}</td>
<td>{{m.mobile}}</td>
</tr>
</table>
```

//pipes



It is used to transform the data before we display it on the HTML
>predefined
>customized

>pedefined
Uppercase
Lowercase
Currency
Json
Percent
…

{{<dataitem>|pipename:<attributes>}}
Ex;
```
textdata="karthik"
salary=1220000
date=new Date()
```
==========

```
<h1>{{textdata|titlecase}}</h1>


<h1>{{textdata|titlecase}}</h1>
```

```
<h1>{{salary|currency:'INR'}}</h1>

<h1>{{date|date:'dd/MM/yy'}}</h1>
```

Customized pipes:

1. define a pipe
2. Define a logic in the pipe for transformation

1. Define a pipe :

```
F:\mphasis angular ws\demoproject>ng g p test
CREATE src/app/test.pipe.spec.ts (179 bytes)
CREATE src/app/test.pipe.ts (213 bytes)
UPDATE src/app/app.module.ts (588 bytes)
```

salary|currency:'INR'

```
@Pipe({
  name: 'test'
})
export class TestPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }

}
```

```
<h1>{{wish|test:4:8}}</h1>

//ts
export class AppComponent {
wish="Good Morning"
}
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'test'
})
export class TestPipe implements PipeTransform {
           //wish           //param1       //param2
  transform(value: string, param1:number,param2:number): string {
    return value.substring(param1,param2);
  }

}
```

Ex-2:

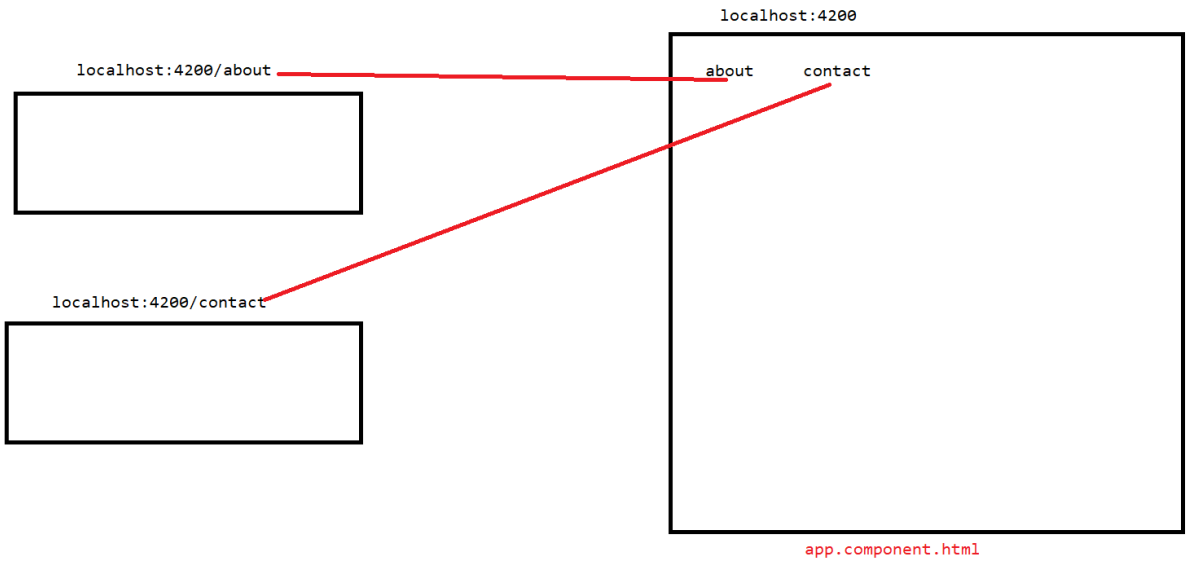
```
<h1>{{person|test:wish}}</h1>
```

```
export class AppComponent {
wish="Good Morning"
person={"name":"bhumi","gender":"f"}


}
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'test'
})
export class TestPipe implements PipeTransform {
           //person           //wish
  transform(value: any, param1:any): string {
   if(value.gender=="m"){
```

```
    return "Hello Mr. "+value.name+"  "+param1
   }
   else{
    return "Hello Miss. "+value.name+"  "+param1
   }
  }

}
```

//Routing :



Hyperlinks are used to navigate from one component to another

1. Config the routes
2. Add the router outlet
3. Add the links in the template

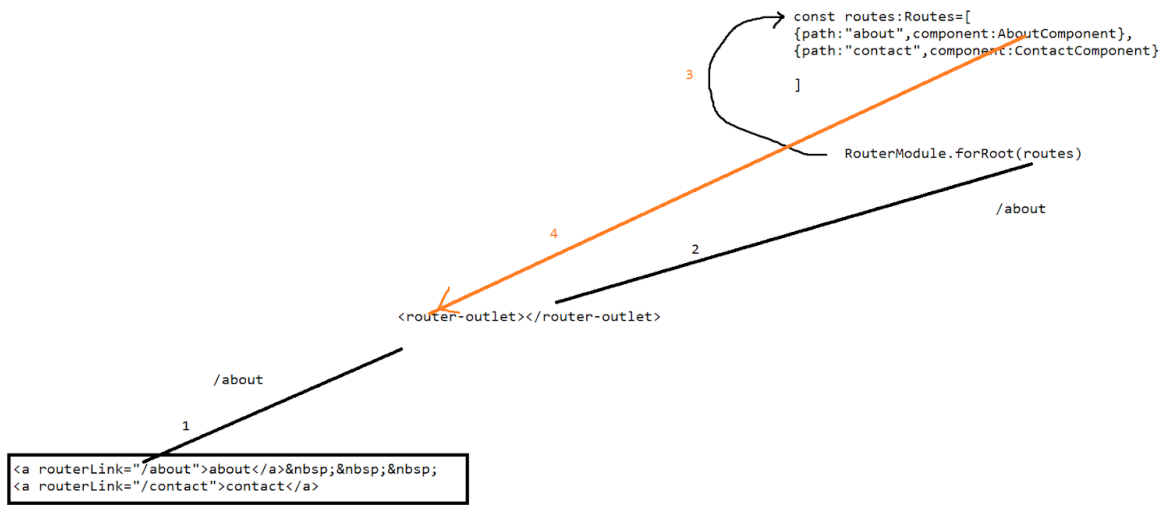
```
F:\mphasis angular ws\demoproject>ng g c about
CREATE src/app/about/about.component.html (20 bytes)
CREATE src/app/about/about.component.spec.ts (592 bytes)
CREATE src/app/about/about.component.ts (271 bytes)
CREATE src/app/about/about.component.css (0 bytes)
UPDATE src/app/app.module.ts (666 bytes)

F:\mphasis angular ws\demoproject>ng g c contact
CREATE src/app/contact/contact.component.html (22 bytes)
CREATE src/app/contact/contact.component.spec.ts (606 bytes)
CREATE src/app/contact/contact.component.ts (279 bytes)
CREATE src/app/contact/contact.component.css (0 bytes)
UPDATE src/app/app.module.ts (752 bytes)
```

>Config the route

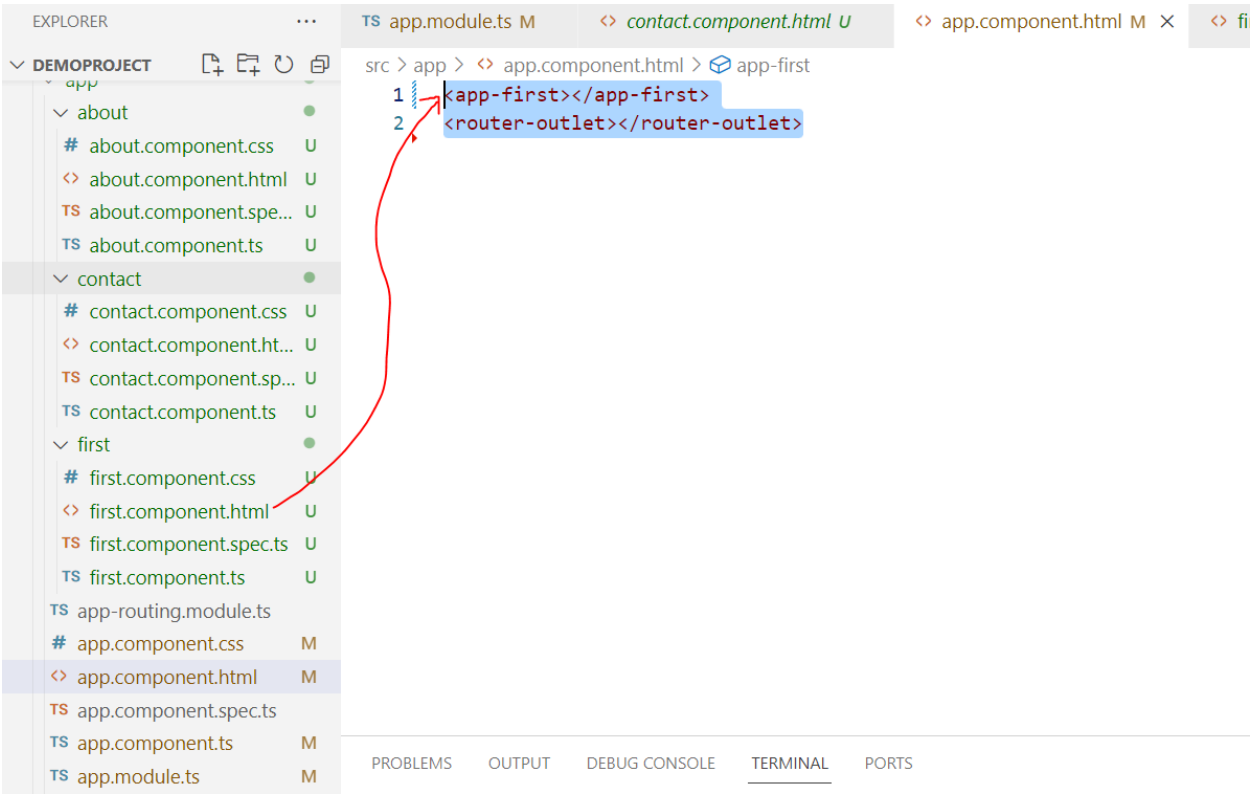
app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FirstComponent } from './first/first.component';
import { FormsModule } from '@angular/forms';
import { TestPipe } from './test.pipe';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';
import { RouterModule, Routes } from '@angular/router';

const routes:Routes=[
{path:"about",component:AboutComponent},
{path:"contact",component:ContactComponent}

]

@NgModule({
  declarations: [
    AppComponent,
```

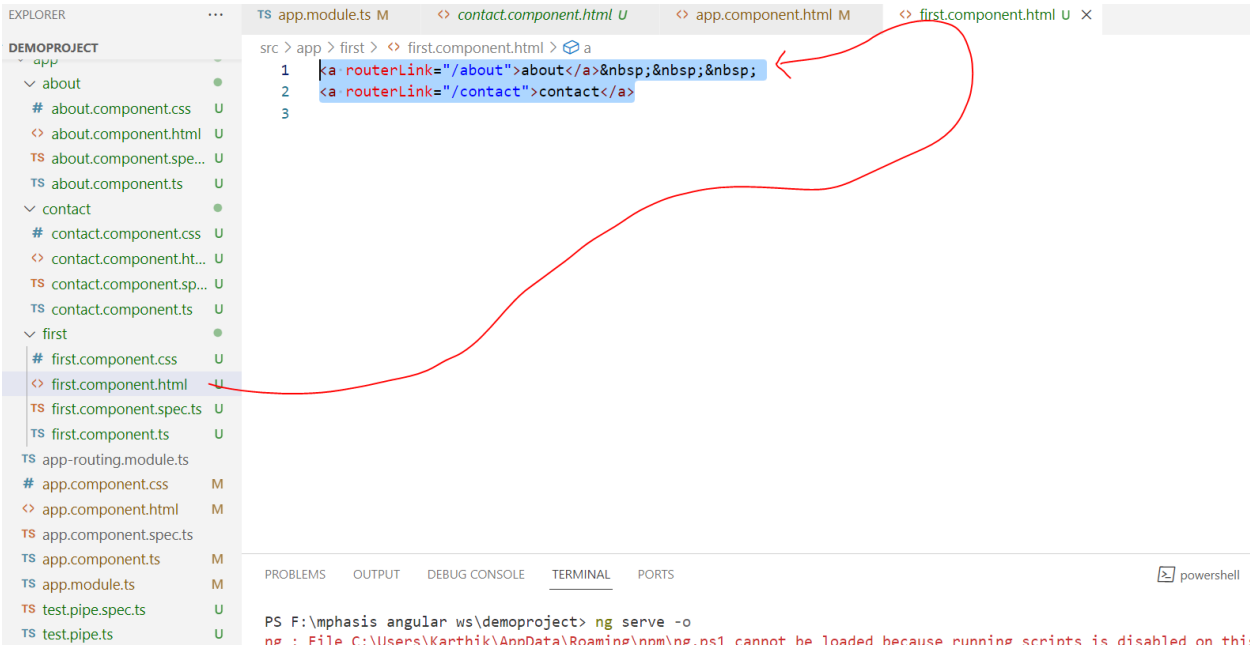
```
    FirstComponent,
    TestPipe,
    AboutComponent,
    ContactComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    RouterModule.forRoot(routes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



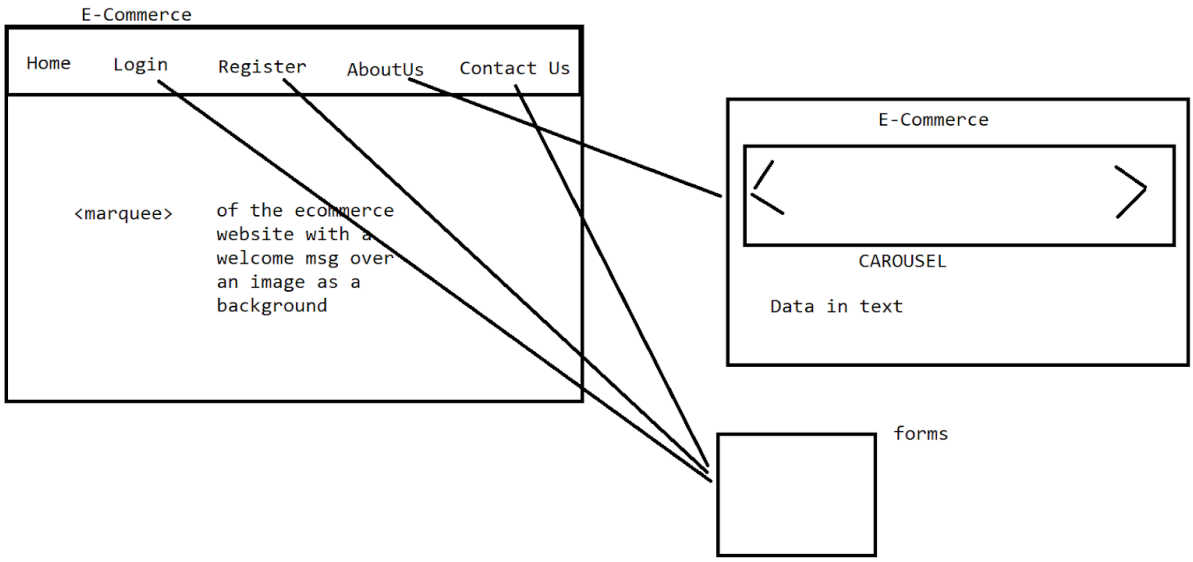
Add the router outlet



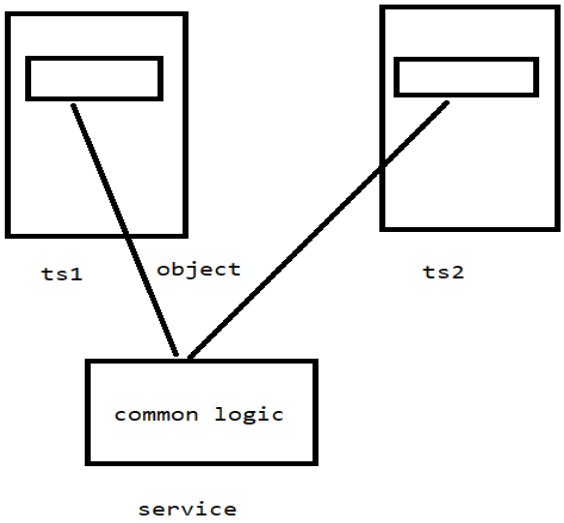
>Define the links



POC



>Service :



Service is the reusable code that is applied on the multiple components .
Rather than we define the config / data on separate TS files and duplicate the LOC . In that situation , we are going to use the service in making that as a common logic

Model -1
We update the TS of the service then the ts of the components which is using that service is going to get the updated logic .

```
F:\mphasis angular ws\demoproject>ng g s servicework
CREATE src/app/servicework.service.spec.ts (382 bytes)
CREATE src/app/servicework.service.ts (140 bytes)
```

```
@Injectable({
  providedIn: 'root'
})
```

This annotation helps in applying changes at the app-root level of the component

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

@Injectable({
  providedIn: 'root'
})
export class ServiceworkService {

  constructor() { }

mobile=['redmi','realme','iphone','samsung']

}
```

//about

```
import { Component, OnInit } from '@angular/core';
import { ServiceworkService } from '../servicework.service';

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

  constructor(private serviceobj:ServiceworkService) { }

  mobiles=this.serviceobj.mobile;
```

```
  ngOnInit(): void {
  }

}
```

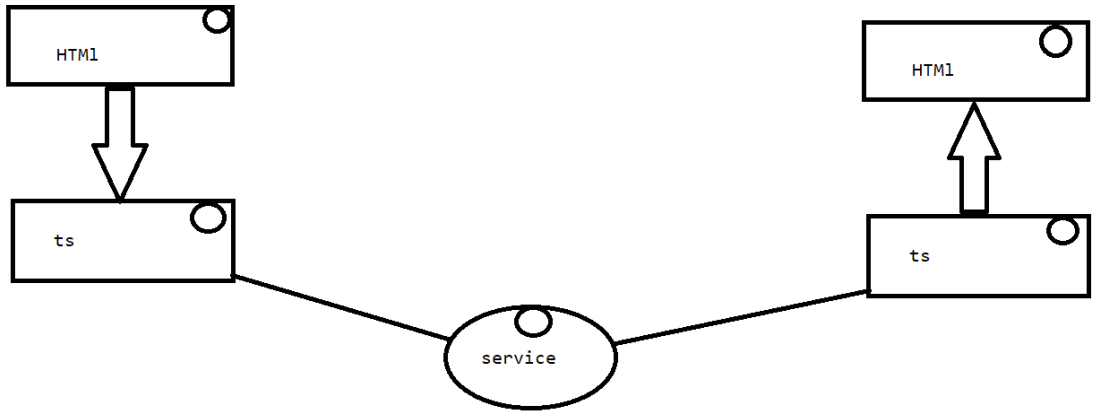
```
<div *ngFor="let m of mobiles">

{{m}}

</div>
```

//contact

Repeat the same as about

Model -2

If the component - html has updated the individual ts file via the update in the service file , then the component that are using this service will be auto updated .




```
<div *ngFor="let m of mobiles">
```

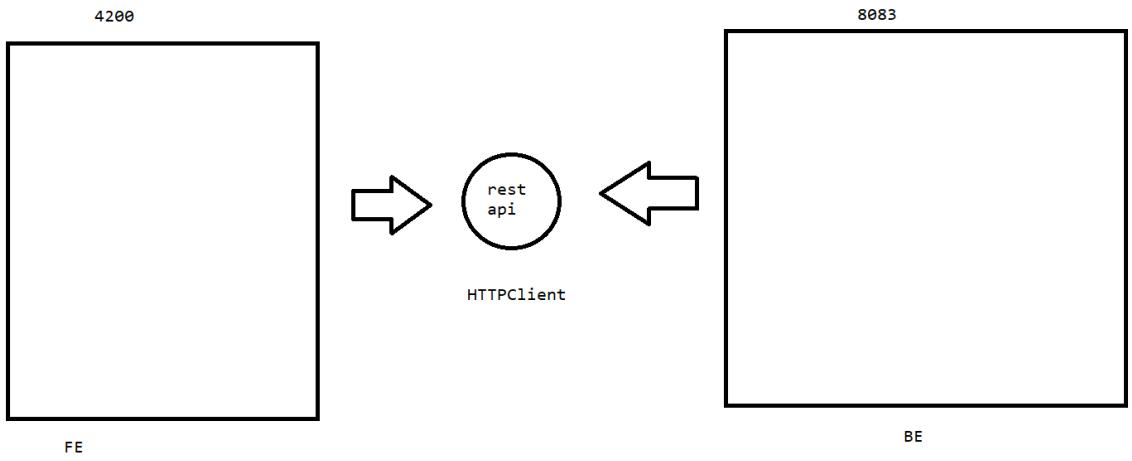
```
{{m}}

</div>
<div>
<button (click)="addmobile()">add mobile</button>
</div>
```

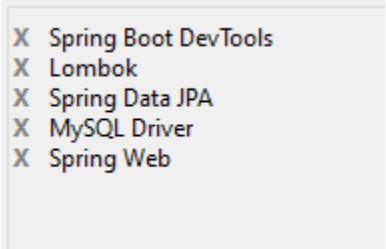
```
import { Component, OnInit } from '@angular/core';
import { ServiceworkService } from '../servicework.service';

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

  constructor(private serviceobj:ServiceworkService) { }

  mobiles=this.serviceobj.mobile;
  ngOnInit(): void {
  }

  addmobile()
  {
    this.serviceobj.mobile.push("oneplus")
  }

}
```

Task :

Take the field of new mobile and add the item to the service

Angular with Spring boot app :






```
package com.example.demo;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@NoArgsConstructor
@AllArgsConstructor
@Data
public class User {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;
        private String name;
```

```
	private String email;
	private int experience;
	private String domain;
}
```

```
package com.example.demo;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepo extends JpaRepository<User, Integer>{

	List<User> findByemail(String email);

}
```

```
package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
@CrossOrigin(origins = "*")//for all external networks we can use hitting this requests
public class UserController {

	@Autowired
	UserRepo repo;
```

```
	//insert
	@PostMapping("/register")
	public String register(@RequestBody User user) {
		repo.save(user);
		return "Hi "+user.getName()+" is registered successfully...!";

	}


	//list of users
	@GetMapping("/getAllusers")
	public List<User> findAllusers(){
		return repo.findAll();
	}


	//delete by id

	@DeleteMapping("/cancel/{id}")
	public List<User> cancelregistration(@PathVariable int id){
		repo.deleteById(id);
		return repo.findAll();
	}


	//search via email
	@GetMapping("/findbyemail/{email}")
	public List<User> findUser(@PathVariable String email){
		return repo.findByemail(email);
	}


//update
}
```

```
#server
server.port=8088

#Jpa hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.show-sql=true
```

spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQLDialect

#datasource
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/db4
spring.datasource.username=root
spring.datasource.password=123456

==========POSTMAN ==========

//Product -> pid , pname , orderdate , cost —---->POJO ===>Back end
POSTMAN -10 min

//Angular

1. Create a new project
2. Create a component and the service
3. Create a class

```
F:\mphasis angular ws>ng new angularspring
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE angularspring/angular.json (2957 bytes)
CREATE angularspring/package.json (1044 bytes)
CREATE angularspring/README.md (1067 bytes)
CREATE angularspring/tsconfig.json (863 bytes)
CREATE angularspring/.editorconfig (274 bytes)
```

=>components and services

```
F:\mphasis angular ws>cd angularspring

F:\mphasis angular ws\angularspring>ng g c registration
? 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
CREATE src/app/registration/registration.component.html (27 bytes)
CREATE src/app/registration/registration.component.spec.ts (641 bytes)
CREATE src/app/registration/registration.component.ts (299 bytes)
CREATE src/app/registration/registration.component.css (0 bytes)
UPDATE src/app/app.module.ts (499 bytes)

F:\mphasis angular ws\angularspring>ng g c searchdelete
CREATE src/app/searchdelete/searchdelete.component.html (27 bytes)
CREATE src/app/searchdelete/searchdelete.component.spec.ts (641 bytes)
CREATE src/app/searchdelete/searchdelete.component.ts (299 bytes)
CREATE src/app/searchdelete/searchdelete.component.css (0 bytes)
UPDATE src/app/app.module.ts (605 bytes)

F:\mphasis angular ws\angularspring>ng g s UserRegisterService
CREATE src/app/user-register-service.service.spec.ts (424 bytes)
CREATE src/app/user-register-service.service.ts (148 bytes)
```

app.module.ts⇒service =>ts =>HTML

Create User.ts

```
export class User{

    name:string;
    email:string;
    experience:number;
    domain:string;
}
```

If you get errors then

tsconfig.json > {} compilerOptions

```
/* To learn more about this file see: https://angular.i
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
    "strict": false,
    "noImplicitOverride": true,
    "noPropertyAccessFromIndexSignature": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2020",
    "module": "es2020",
    "lib": [
      "es2020",
      "dom"
    ]
  },
  "angularCompilerOptions": {
    "enableI18nLegacyMessageIdFormat": false,
    "strictInjectionParameters": true,
    "strictInputAccessModifiers": true,
    "strictTemplates": true
```

app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { RegistrationComponent } from
'./registration/registration.component';
import { SearchdeleteComponent } from
'./searchdelete/searchdelete.component';
import { RouterModule, Routes } from '@angular/router';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';


const routes: Routes = [
{path:"",component:RegistrationComponent},
{path:"register",component:RegistrationComponent},
{path:"search",component:SearchdeleteComponent}
];

@NgModule({
  declarations: [
    AppComponent,
    RegistrationComponent,
    SearchdeleteComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
    RouterModule.forRoot(routes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Service

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

@Injectable({
  providedIn: 'root'
})
export class UserRegisterServiceService {

  constructor(private http:HttpClient) { }

  public doregistration(user:any){
    return
this.http.post("http://localhost:8088/register",user,{responseType:'text'
as 'json'});
  }

  public getusers(){
    return this.http.get("http://localhost:8088/getAllusers");
  }


  public getuserbyemail(email:any){
return this.http.get("http://localhost:8088/findbyemail/"+email);
  }


  public deletebyid(id:any){
    return this.http.delete("http://localhost:8088/cancel/"+id);
  }

}
```

//registration

```
<h1>{{message}}</h1>

<form>
Name<input type="text" name="name" [(ngModel)]="user.name"><br>
```

```
Email<input type="email" name="email" [(ngModel)]="user.email"><br>
Experience<input type="text" name="experience"
[(ngModel)]="user.experience"><br>
Domain<input type="text" name="domain" [(ngModel)]="user.domain"><br>
<input type="submit" value="register" (click)="registerNow()"><br>
</form>


import { Component, OnInit } from '@angular/core';
import { UserRegisterServiceService } from
'../user-register-service.service';
import { User } from '../User';

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

user:User=new User();
message=""
  constructor(private service:UserRegisterServiceService) { }

  ngOnInit(): void {
  }

  public registerNow(){
                let response=this.service.doregistration(this.user);
                response.subscribe((data:any)=>this.message=data)
  }

}
```

//search-delete

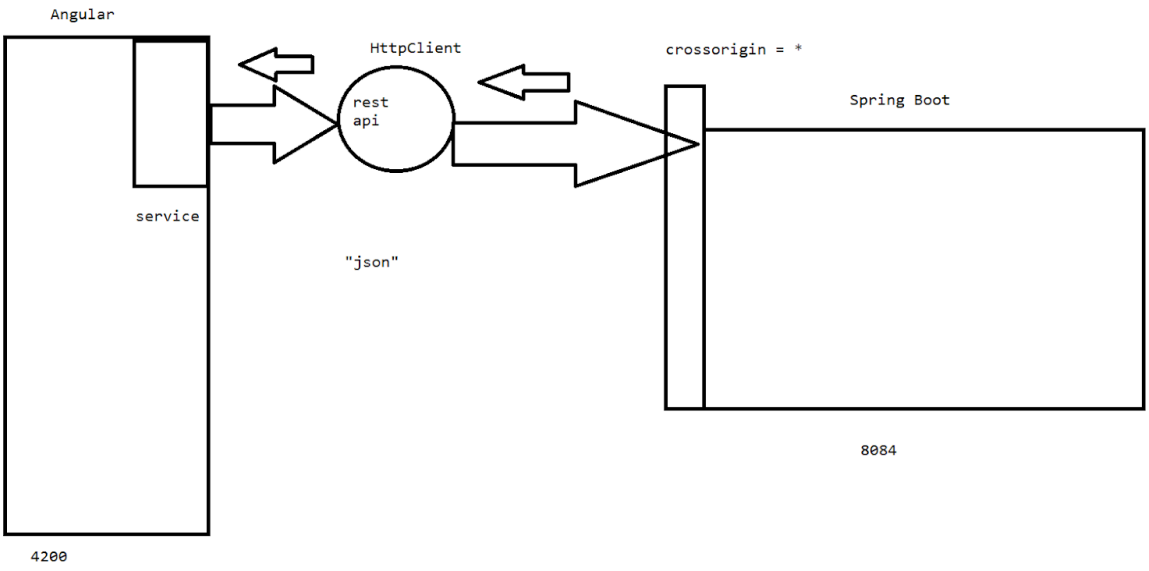
```
<br>
<br>

Enter an email to search <input type="text" name="email"
[(ngModel)]="email"><button (click)="finduserbyemail()">search</button>
<h1><i>List of users with experience and domain</i></h1>

<table border="1">
<tr><th>Id</th><th>Name</th><th>Email</th><th>Experience</th><th>Domain</t
h></tr>
<tr *ngFor="let user of users">
<td>{{user.id}}</td>
<td>{{user.name}}</td>
<td>{{user.email}}</td>
<td>{{user.experience}}</td>
<td>{{user.domain}}</td>
<td><button (click)="deleteuser(user.id)">Delete</button></td>
</tr>

</table>
```

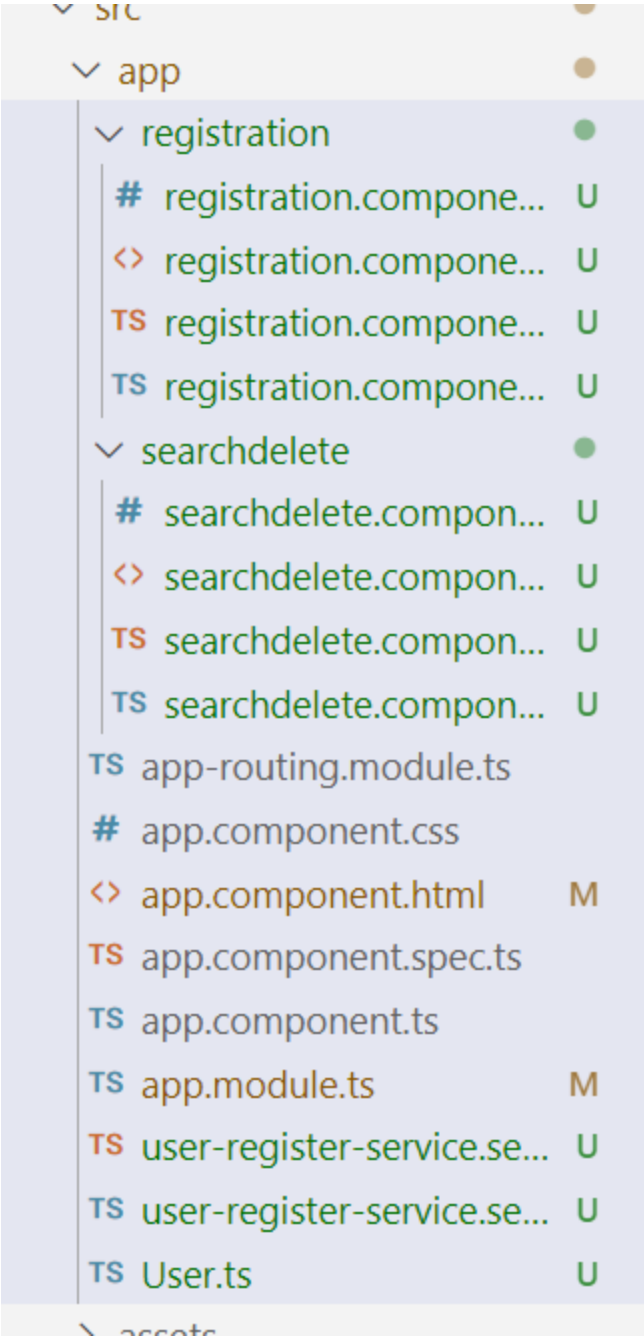
```
import { Component, OnInit } from '@angular/core';
import { UserRegisterServiceService } from
'../user-register-service.service';

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

  constructor(private service:UserRegisterServiceService) { }
email=""
users:any
```

```
//by default if we dont need any function to be called then we need to
write the code in the below block
  ngOnInit(): void {
    let response=this.service.getusers()
    response.subscribe((data:any)=>this.users=data)
  }
  finduserbyemail(){
    let response=this.service.getuserbyemail(this.email)
    response.subscribe((data:any)=>this.users=data)
  }
  deleteuser(id:number){
    let response=this.service.deletebyid(id);
    response.subscribe((data:any)=>this.users=data)
  }

}
```

App.component.html

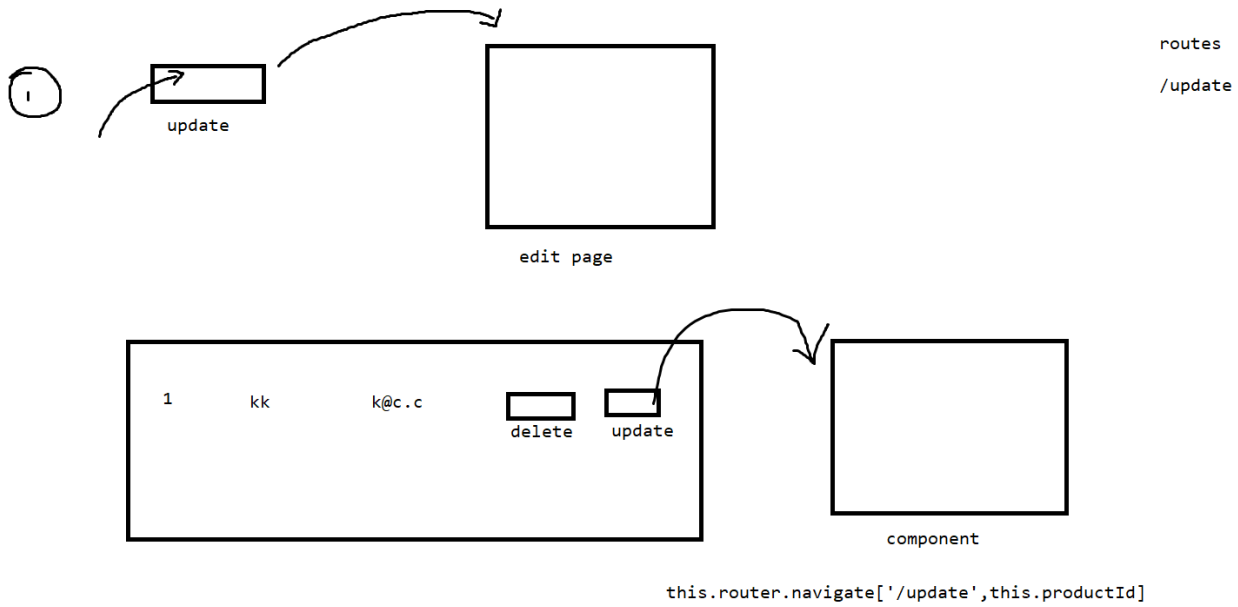
```
<a routerLink="/register">register</a>   
<a routerLink="/search">search</a>
<router-outlet></router-outlet>
```


Angular
HttpClient
crossorigin = *
rest
api
Spring Boot
service
"json"
8084
4200



For the above : update

For product perform all CRUD operations via postman



<button routerLink="/edit/{{product.id}}">edit</button>

Step -1

```
<td>{{user.email}}</td>
<td>{{user.experience}}</td>
<td>{{user.domain}}</td>
<td><button (click)="deleteuser(user.id)">Delete</button></td>
<td><button routerLink="/update/{{user.id}}" >Update</button></td>
</tr>
```

Step-2

```
Go to component
<h1>{{message}}</h1>


<form>
 id<input type="number" name="id" [(ngModel)]="user.id"><br>
Name<input type="text" name="name" [(ngModel)]="user.name"><br>
Email<input type="email" name="email" [(ngModel)]="user.email"><br>
Experience<input type="text" name="experience" [(ngModel)]="user.experience"><br>
Domain<input type="text" name="domain" [(ngModel)]="user.domain"><br>
<input type="submit" value="register" (click)="update()"><br>
</form>
```

Step -3 –ts

```
  public update(){
    console.log(this.user.id)
                let response=this.service.update(this.user);
                response.subscribe((data:any)=>this.message=data)
  }
}
```

Step-4-service

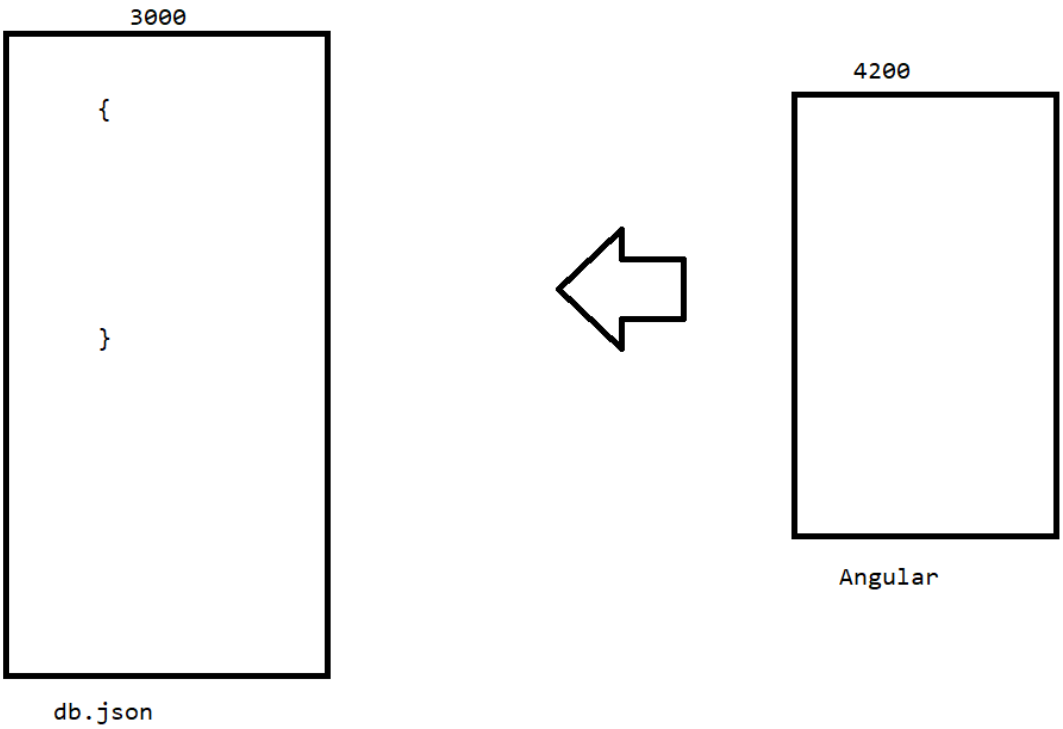
```
public update(user:any){
  console.log(user.domain)
  return this.http.put("http://localhost:8088/update",user,{responseType:'text' as 'json'});
}
```

Step -5 -spring boot

```
//update

    //update
    @PutMapping("/update")
    public String update(@RequestBody User user) {
      //find the user by id
      log.info("request hit"+user.id);
      User existingUser = repo.findById(user.id).orElse(null);
      //check if the user exists
      if (existingUser != null) {
        //set the new values for the user fields

        existingUser.setName(user.getName());
        existingUser.setEmail(user.getEmail());
        existingUser.setExperience(user.getExperience());
        existingUser.setDomain(user.getDomain());
        //save the updated user in the database
        repo.save(existingUser);
        //return a success message
        return "Hi " + user.getName() + " is updated successfully...!";
      } else {
        //return an error message
        return "User not found with id " + user.id;
      }
    }
```

//debugging
Angular - console.log
Spring boot - log.info

–Json-server



It is a json db server that can be manipulated according to the crud operations.

–install the json server

```
F:\>npm install -g json-server

added 3 packages, removed 202 packages, changed 113 packages, and audited 117 packages in 14s

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

found 0 vulnerabilities
```

-db.json

```
{
"students":[
{
"id":1,
"name":"ravi"
},

{
```

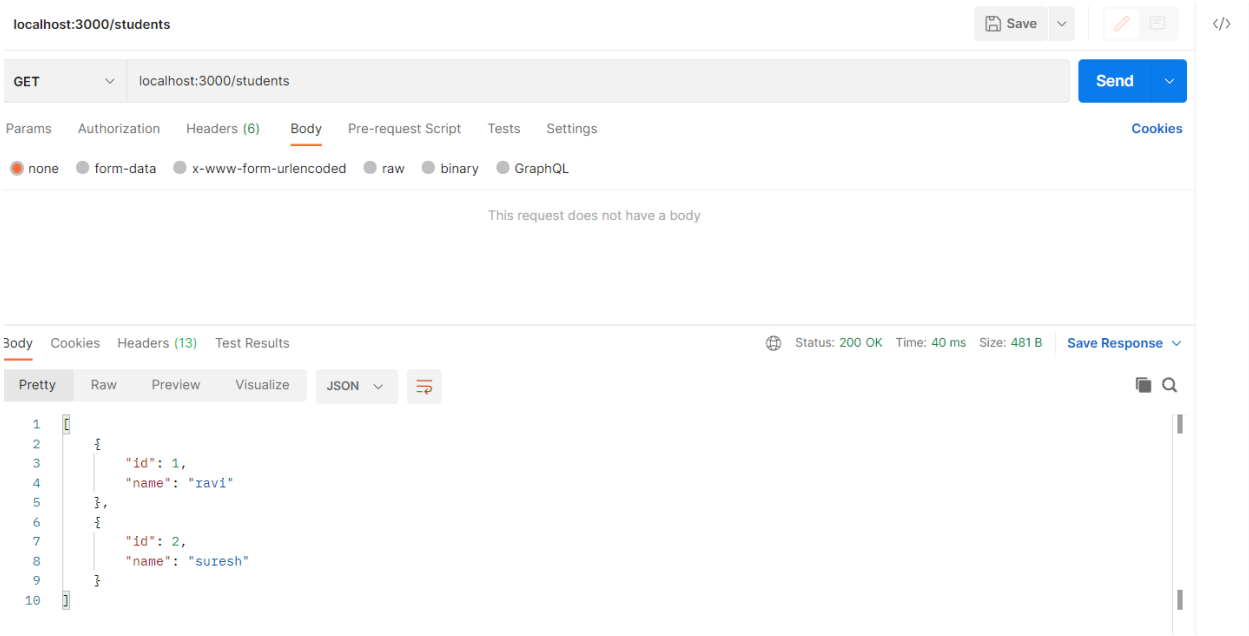
"id":2,
"name":"suresh"
}
]

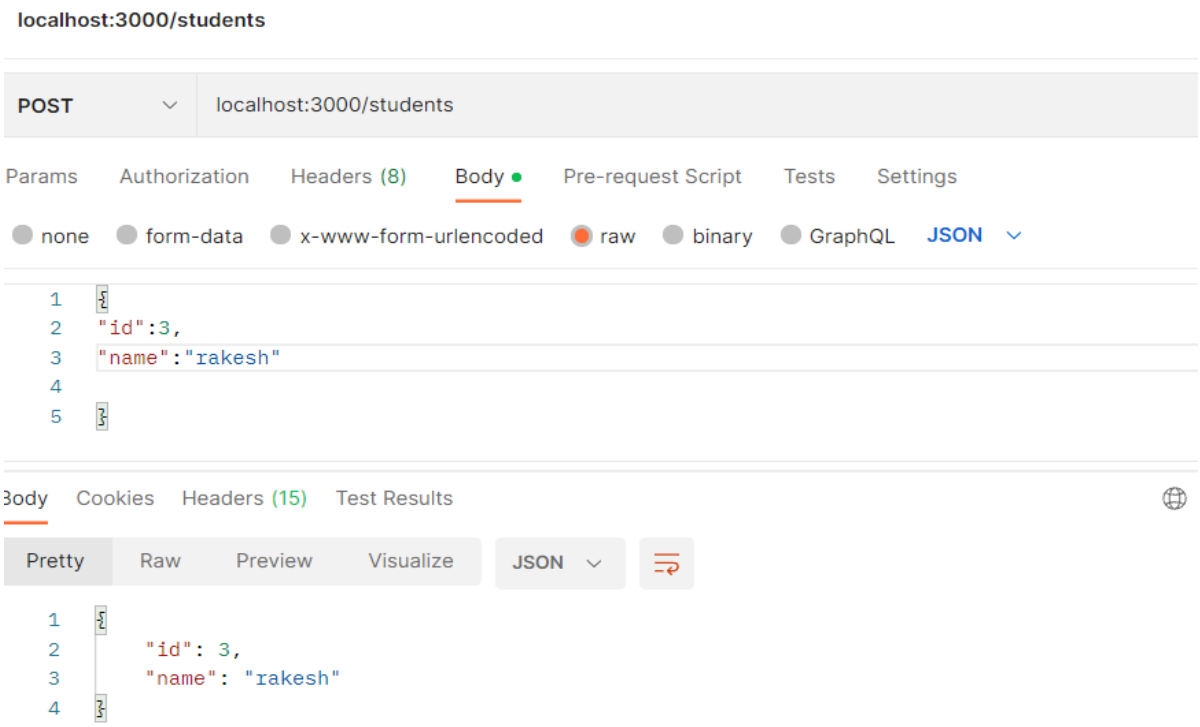
}

–start the server

```
F:\>json-server --watch db.json

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  http://localhost:3000/students

  Home
  http://localhost:3000

  Type s + enter at any time to create a snapshot of the database
  Watching...

GET /students 200 34.240 ms - 86
```

Operate with postman





Overview | GET localhost:3000/studen

localhost:3000/students/1

GET localhost:3000/students/1

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
1 {
2 "id":3,
3 "name":"rakesh"
4
5 }
```

Body | Cookies | Headers (13) | Test Results | Status: 200 OK | Tim

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2     "id": 1,
3     "name": "ravi"
4 }
```

Overview | DEL localhost:3000/studen

localhost:3000/students/1

DELETE localhost:3000/students/1

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
1 {
2 "id":3,
3 "name":"rakesh"
4
5 }
```

Body | Cookies | Headers (13) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
1 {}
```

Overview | PUT localhost:3000/studen | No Envir

localhost:3000/students/2 | Sav

PUT localhost:3000/students/2

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
1 {
2 "name":"gaurav"
3 }
```

Body | Cookies | Headers (13) | Test Results | Status: 200 OK | Time: 41 ms | Size: 428

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2     "name": "gaurav",
3     "id": 2
4 }
```

=>create angular with json server

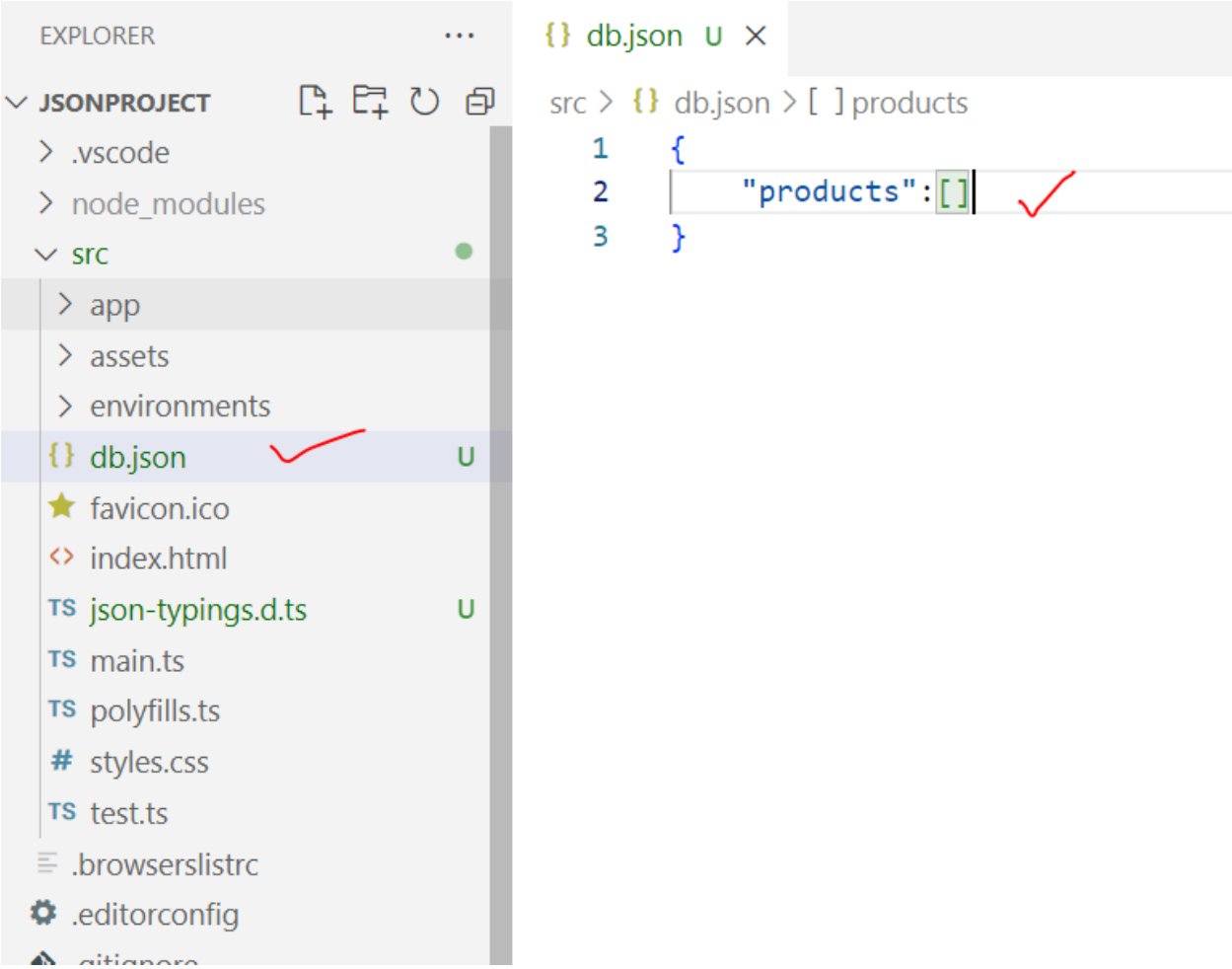
```
(c) Microsoft Corporation. All rights reserved.

F:\mphasis angular ws>ng new jsonproject
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE jsonproject/angular.json (2947 bytes)
CREATE jsonproject/package.json (1042 bytes)
CREATE jsonproject/README.md (1065 bytes)
CREATE jsonproject/tsconfig.json (863 bytes)
```

1. Load the project into the Vs
2. Config the below file

TS json-typings.d.ts U

```
declare module "*.json" {
    const value: any;
    export default value;
  }
```

Create a json db



Start the json server

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

F:\mphasis angular ws\jsonproject\src>json-server --watch db.json

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  http://localhost:3000/products

  Home
  http://localhost:3000

  Type s + enter at any time to create a snapshot of the database
  Watching...
```

```
F:\mphasis angular ws\jsonproject>ng g c productlist
? 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
CREATE src/app/productlist/productlist.component.html (26 bytes)
CREATE src/app/productlist/productlist.component.spec.ts (634 bytes)
CREATE src/app/productlist/productlist.component.ts (295 bytes)
CREATE src/app/productlist/productlist.component.css (0 bytes)
UPDATE src/app/app.module.ts (495 bytes)

F:\mphasis angular ws\jsonproject>ng g s productservice
CREATE src/app/productservice.service.spec.ts (397 bytes)
CREATE src/app/productservice.service.ts (143 bytes)
```

App.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ProductlistComponent } from
'./productlist/productlist.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { RouterModule, Routes } from '@angular/router';
import { HttpClientModule } from '@angular/common/http'
const routes:Routes=[
  {path:"",component:ProductlistComponent},
  {path:"products",component:ProductlistComponent}
  ]

@NgModule({
  declarations: [
    AppComponent,
    ProductlistComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
    ReactiveFormsModule,
    RouterModule.forRoot(routes)

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

//component

```
<ul>
    <li *ngFor="let  product of products">
    {{product.name}}-----{{product.description}}
    </li>
    </ul>



    <h2>Add product</h2>
    <div>
    Name: <input type="text" id="name" [(ngModel)]="name">
    description:<input type="text" id="description"
[(ngModel)]="description">
    <button (click)="addproduct(name,description)">Add Product</button>
    </div>
```

Ts:

```
import { Component, OnInit } from '@angular/core';
import { ProductserviceService } from '../productservice.service';

@Component({
  selector: 'app-productlist',
  templateUrl: './productlist.component.html',
  styleUrls: ['./productlist.component.css']
})
export class ProductlistComponent implements OnInit {
products: any[]=[];
id=""
name=""
description=""

  constructor(private productservice:ProductserviceService) { }

  ngOnInit(): void {
    this.getProducts();
  }
```

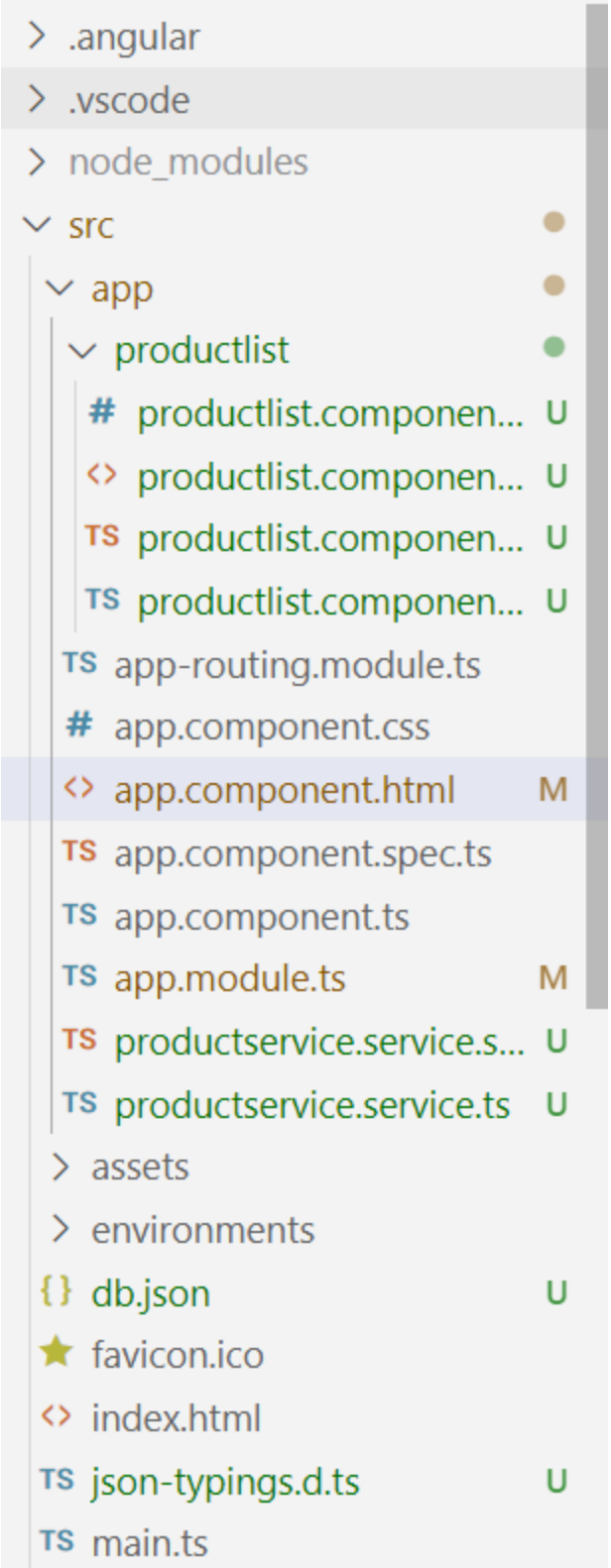
```
  addproduct(name:string,description:string){
    name=name.trim();
    description=description.trim();
    const product={name,description};

this.productservice.addproduct(product).subscribe(newProduct=>{this.produc
ts.push(newProduct)})
  }

getProducts():void{
  this.productservice.getproducts().subscribe(data=>{this.products=data});
}

}
```

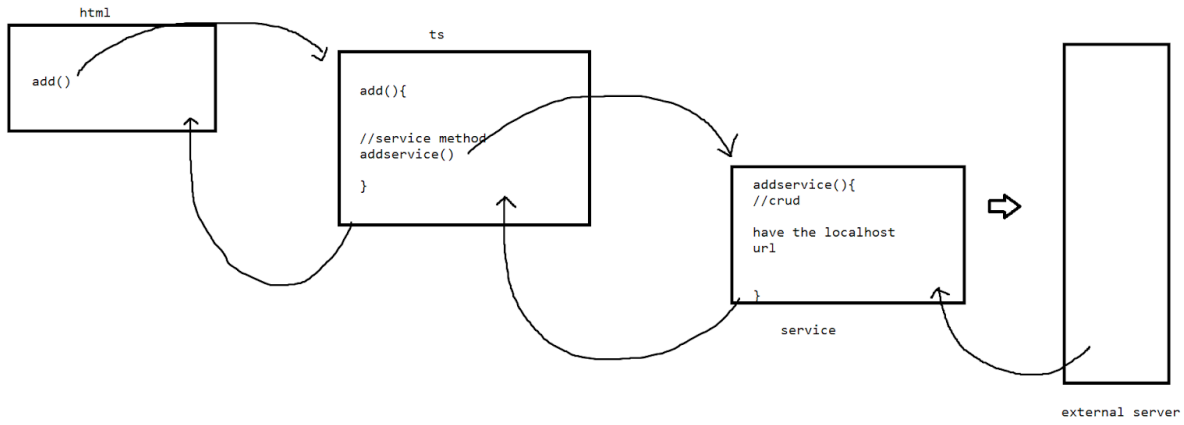
Service

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ProductserviceService {
private url='http://localhost:3000/products';
  constructor(private http:HttpClient) { }

addproduct(product:any){
  return this.http.post(this.url,product);
}

getproducts():Observable<any>{
  return this.http.get(this.url);
}

}
```

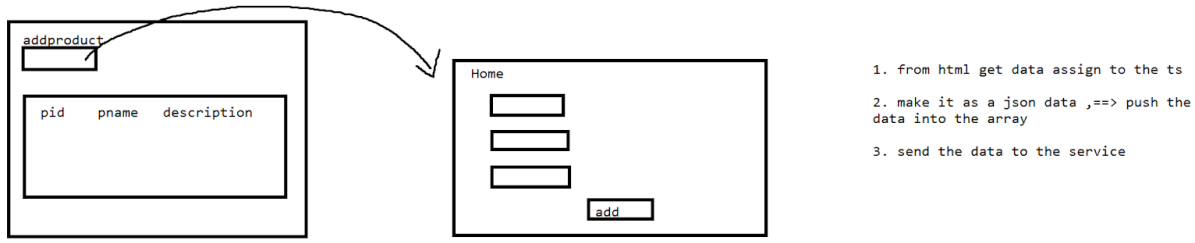
app.module.ts

```
<app-productlist></app-productlist>
```


.angular
.vscode
node_modules
src
app
productlist
productlist.componen... U
productlist.componen... U
productlist.componen... U
productlist.componen... U
app-routing.module.ts
app.component.css
app.component.html M
app.component.spec.ts
app.component.ts
app.module.ts M
productservice.service.s... U
productservice.service.ts U
assets
environments
db.json U
favicon.ico
index.html
json-typings.d.ts U
main.ts



//product - edit and delete



For all the services use ->Observable<any> ===> null,single object ,list of objects

```
deleteuser(id:number){
  let response=this.productservice.deletebyid(id);
  response.subscribe(()=>{this.products=this.products.filter(product=>product.id!==id)});

}
```

```
updateproduct: any = null;
products: any[]=[];
name = ''; // Initialize with current product name
description = ''; // Initialize with current product description
productId = ''; // Initialize with the ID of the product being upd

ngOnInit(): void {

  this.route.paramMap.subscribe(paramas=>{
    const idParam = paramas.get('id');
    console.log(idParam);
    this.productId+=idParam
    console.log(this.productId);
  })

}
```

//update

```
const product={

name:this.name,
description:this.description


}
```

arg

(this.id,product)

service

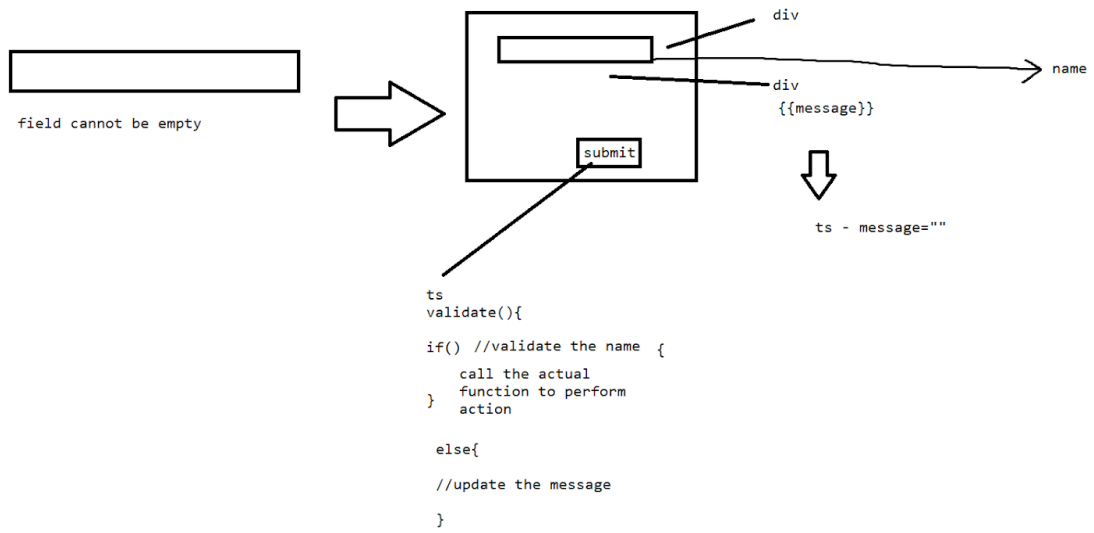
id

localhost:3000/3 , product

1. Product - Components -> Add product ->product page
2. Products -> tabular content -> edit and delete

==========

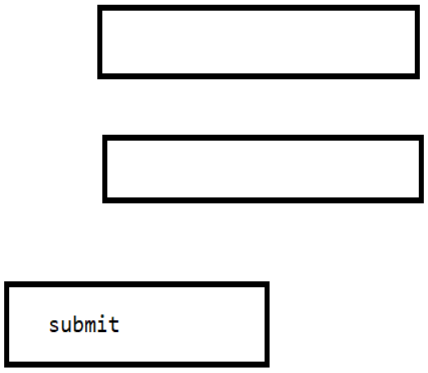
Angular validation



Step -1
Create a field and have ngModel mapping to the ts file , for each field have a for reflecting the messages
Step-2
Submit the form by (click)= with some function name , ex:validate()
Step-3
In validate function the ngModel mapped data is validated , if true we allow to the actual function , else we update the messages fields of the div .

{{message}}

Task : 10 min



Validate 2 fields .

//Project

Step 1

Create a json data as below

```
{
   "employees": [
     {
       "id": 1,
       "first_name": "Sebastian",
       "last_name": "Eschweiler",
       "email": "sebastian@codingthesmartway.com"
     },
     {
       "id": 2,
       "first_name": "Steve",
       "last_name": "Palmer",
       "email": "steve@codingthesmartway.com"
     },
     {
       "id": 3,
       "first_name": "Ann",
       "last_name": "Smith",
       "email": "ann@codingthesmartway.com"
     }
   ]
 }
```

Start your json server

Step -2
Create the components and services

Step- 3
Complete the service functions of the crud

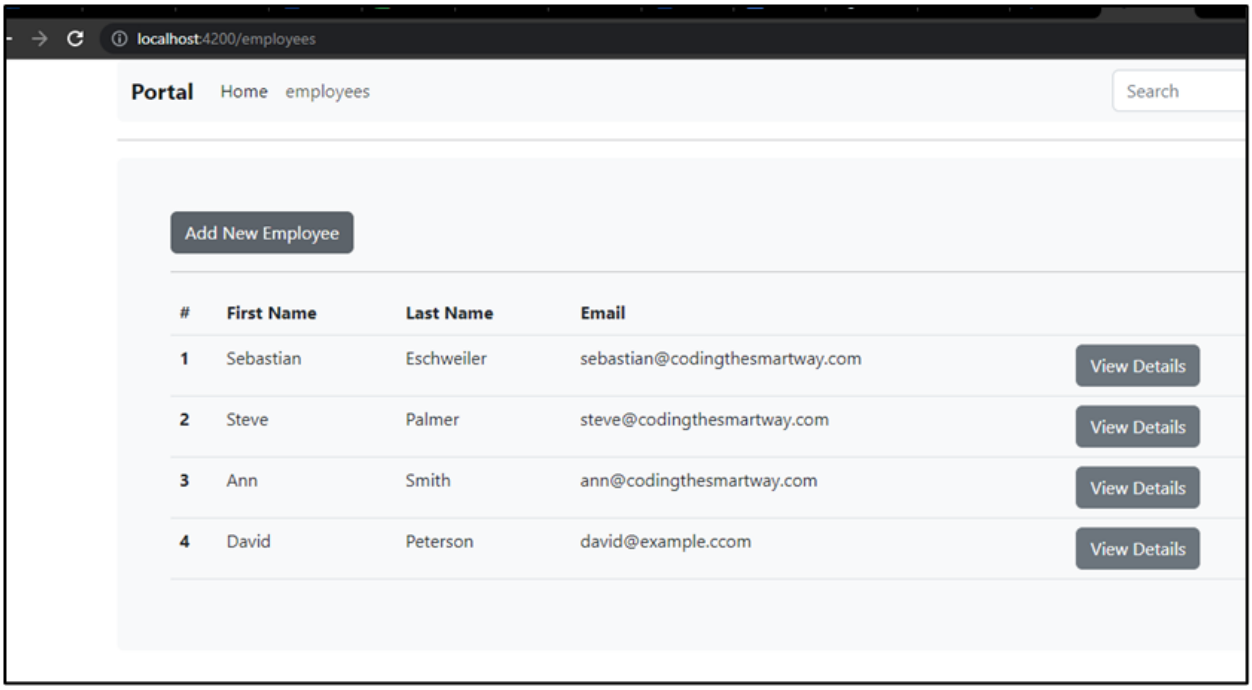
Step -4

Config the app.module.ts

Step-5
Go to components and take the field data from the forms and map it with the TS and call the service functions

Functional side:

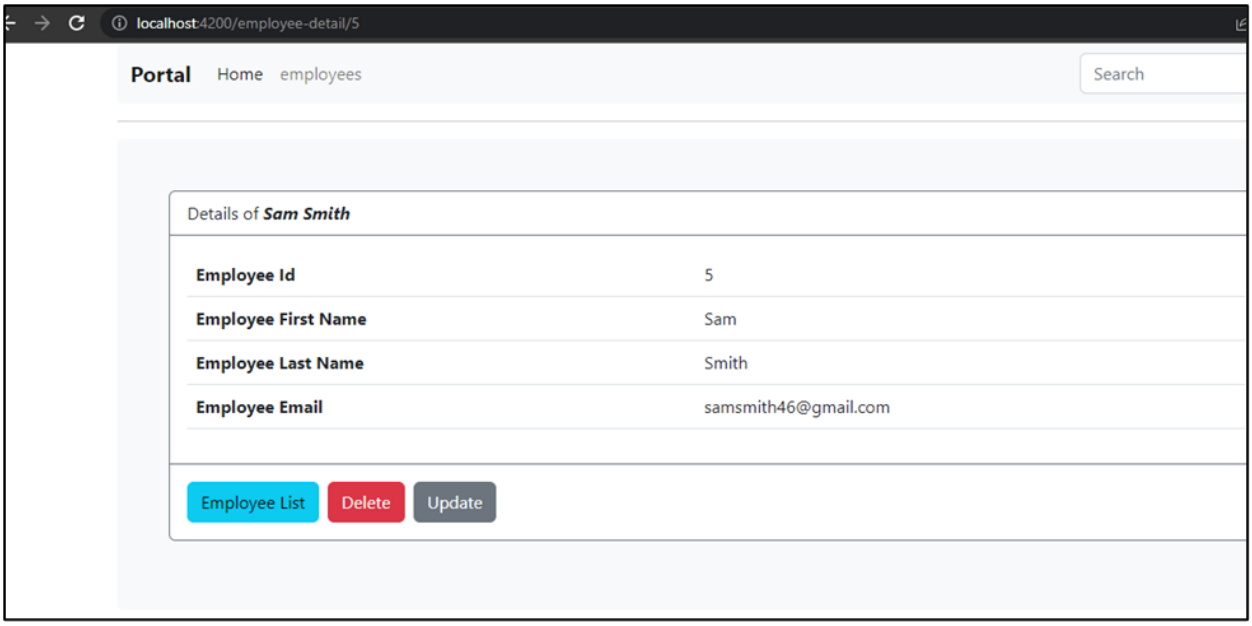
1. Have admin login –hardcoding or with the json file maintained for this
2. Once entered have a HOME that has the add button and view details button displaying on the UI in tabular content => boot strap




```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody>
```

```
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td colspan="2">Larry the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

View details has → for buttons bootstrap colors



```
import {Router} from ‘@angular/router’

constructor (private router:Router)

this.router.navigate([‘view’]);===>component path that config in routes
```