**Passing Parameters to Route**

Here we will discuss routing parameter, here we will see how data can also be passed from one component to other with the help of routing parameter.

So, first question arise is, what is routing parameter?

Yesterday, we have configured routes. So, there we have set path in route array.

Example:

Localhost://4200/regform

But let’s consider that we have a component as an user. There will be multiple users. So, We want to access the user whose id=20 then it’s approximate url will be like

**Localhost://4200/user/20**

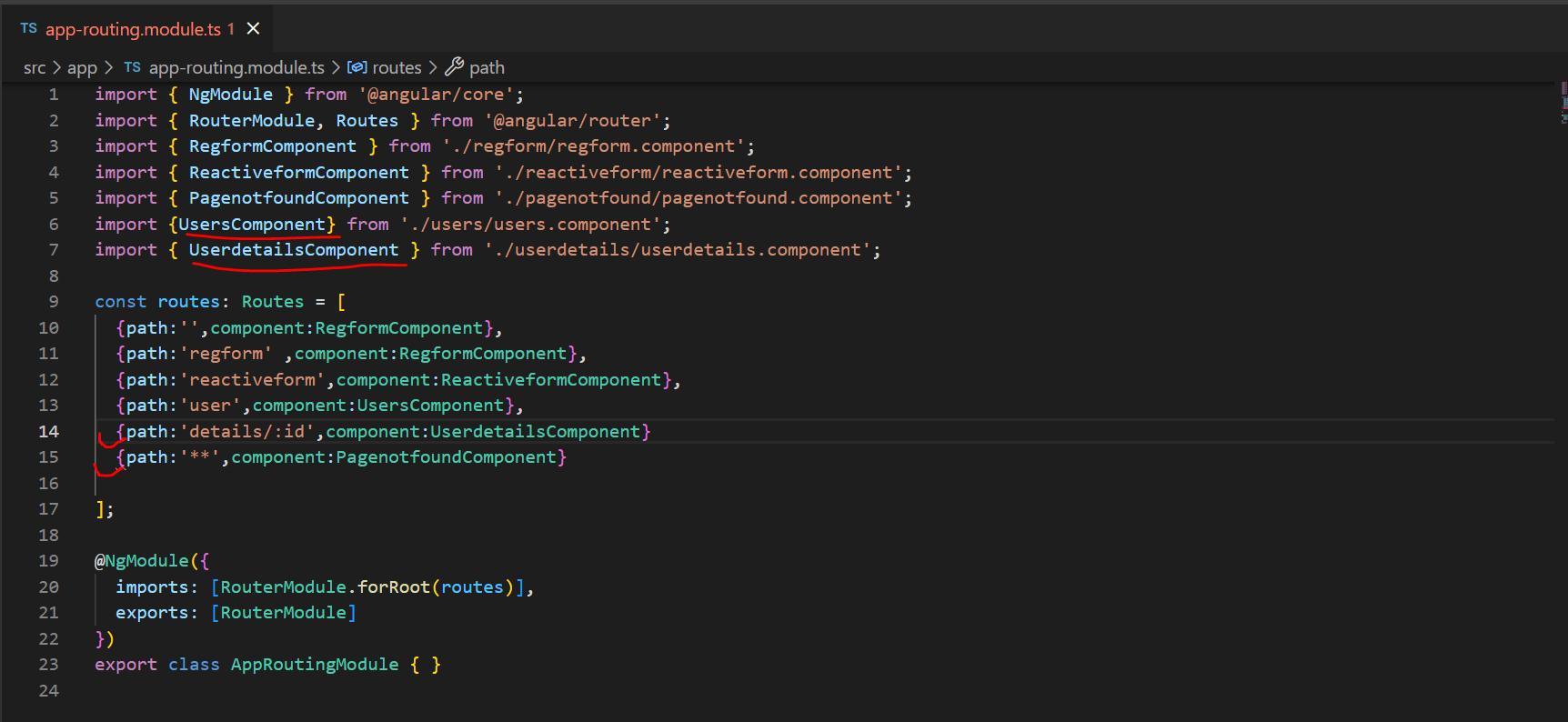
So, here this route is considered as route with parameter.

For implementing this we will see a simple demo.

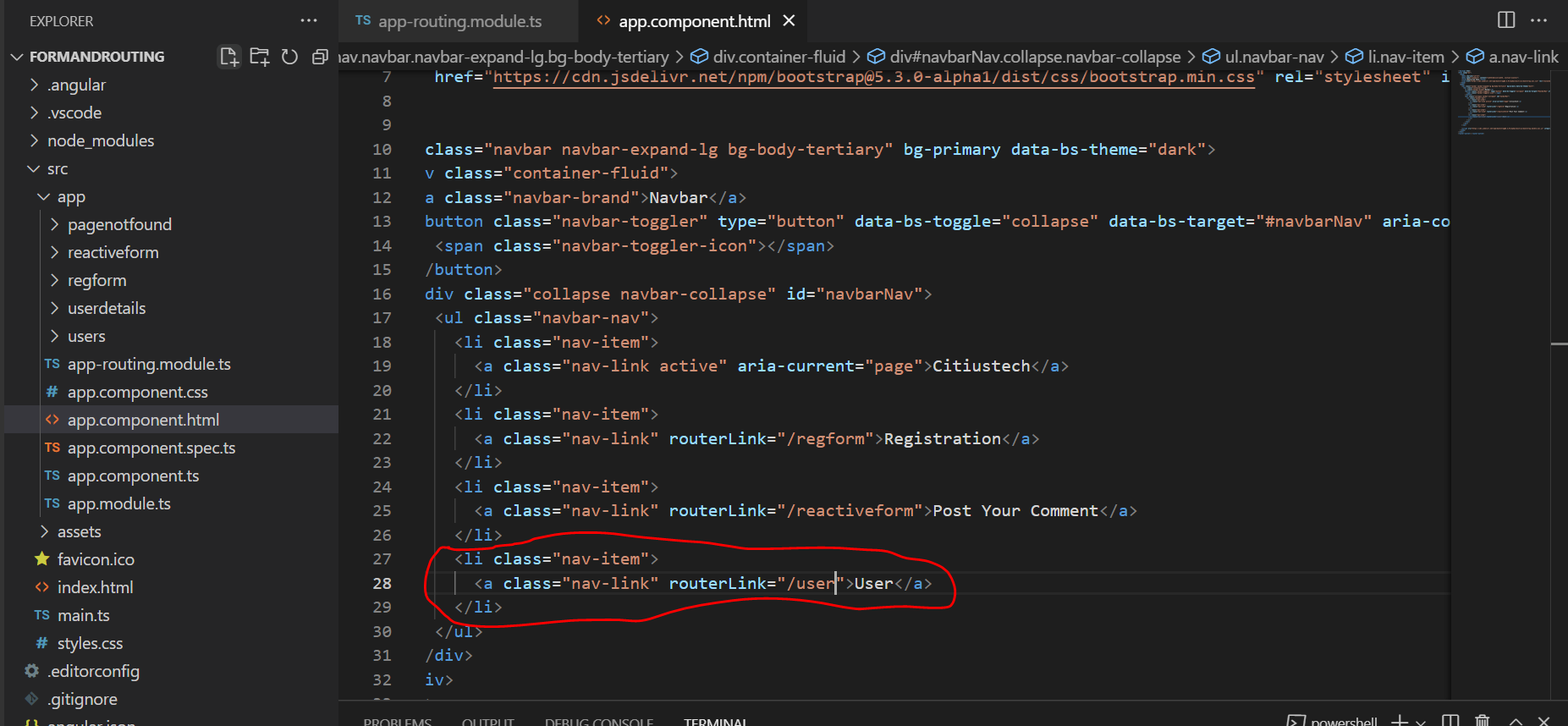
Add two components

1. ng g c users
2. ng g c userdetails

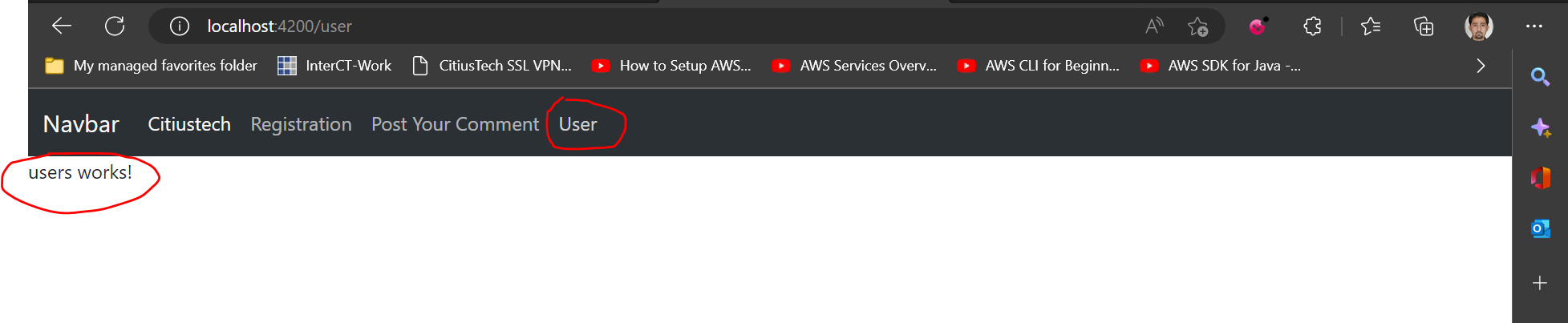
Now, import both the components app-routing.module.ts file.



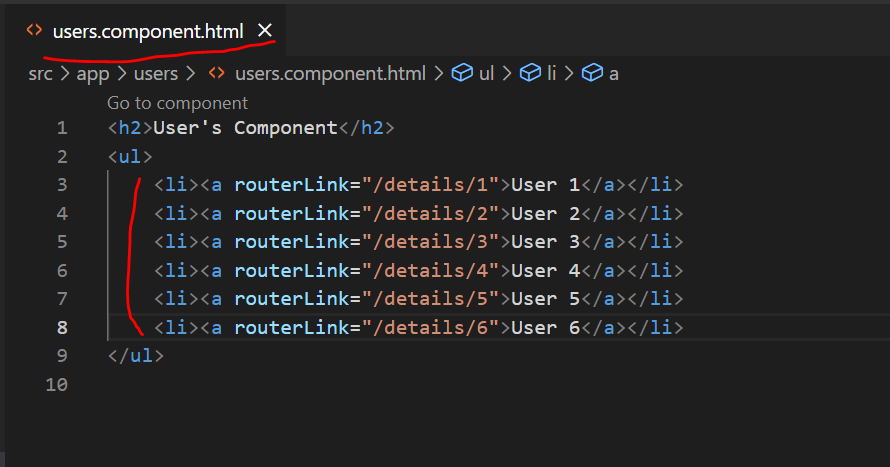
Open, app.component.html and add routerLink for these components.



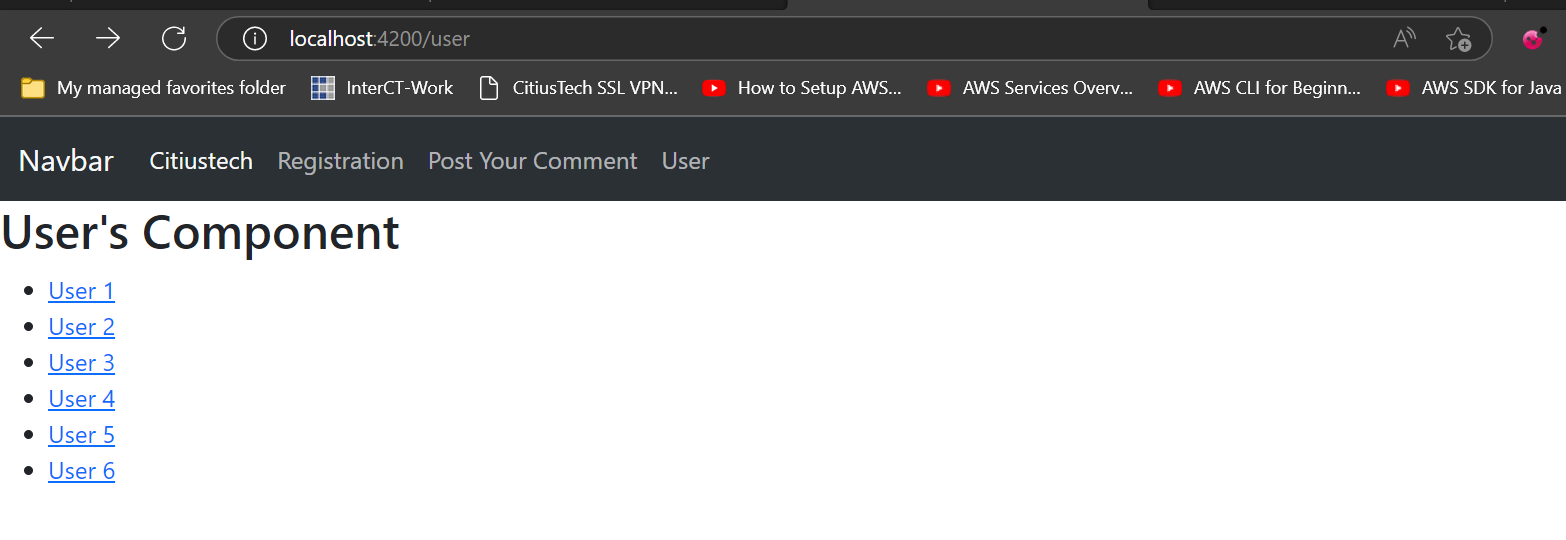
User component is routed properly and it’s working.



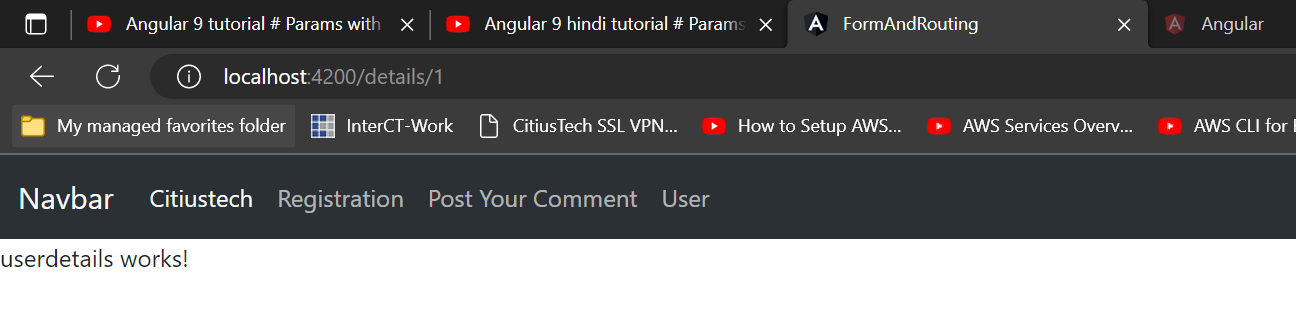
Now, let’s create some links on user.component.html



Output:



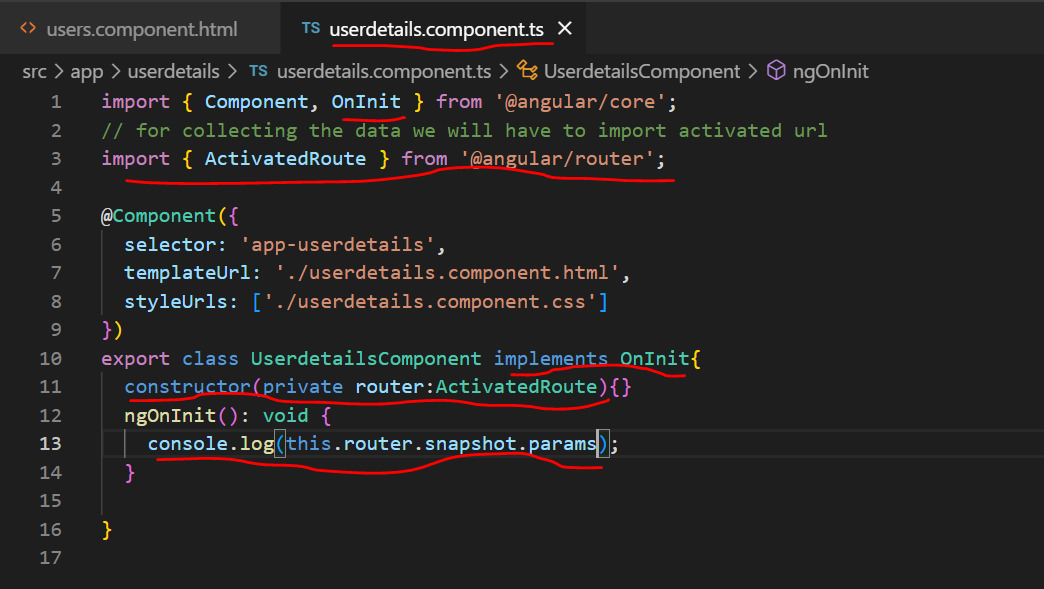
If click on any of link it will route to user-details component.



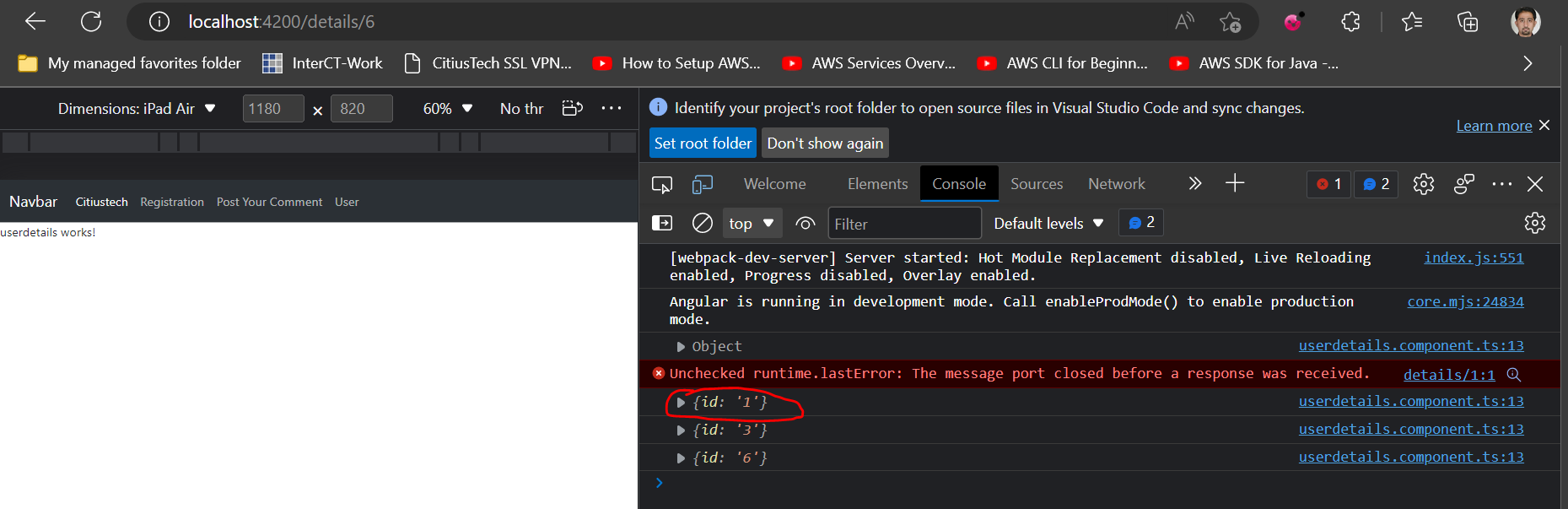
Now. Let’s collect the data in userdetail component.

For this purpose, open userdetail.component.ts file

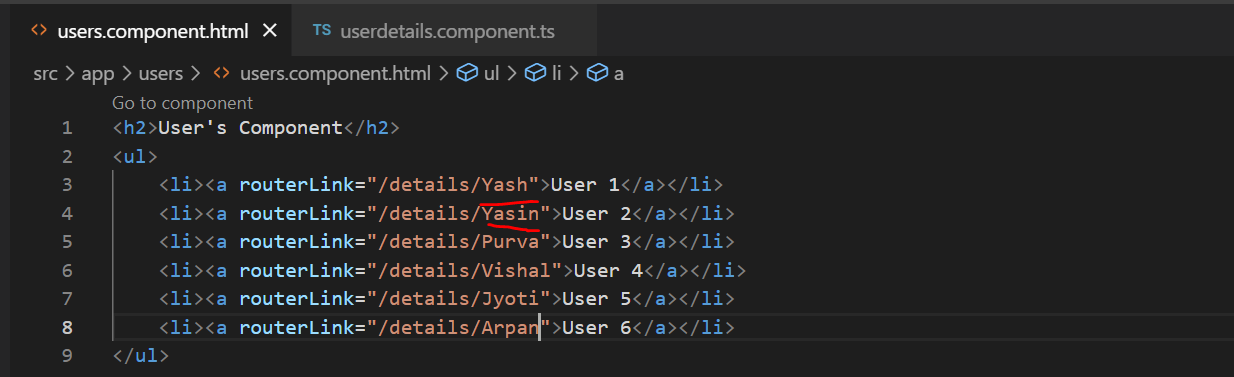
[Angular - ActivatedRouteSnapshot](https://angular.io/api/router/ActivatedRouteSnapshot)



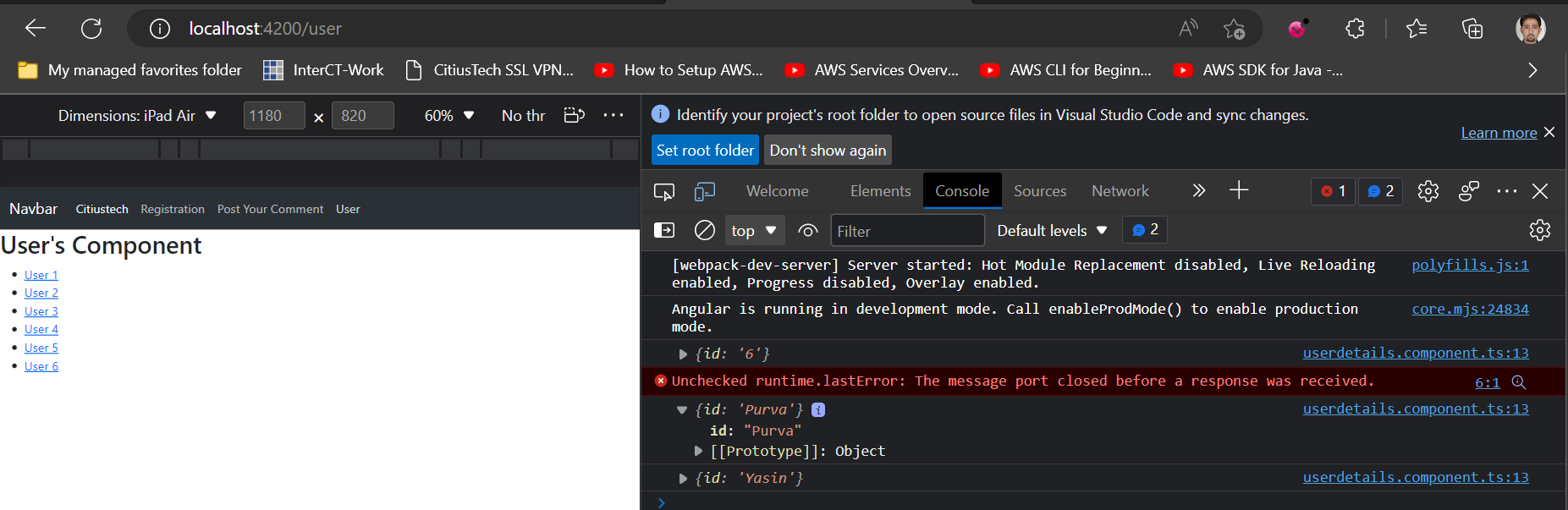
This will log the id of user in console.



We can also pass name as well.

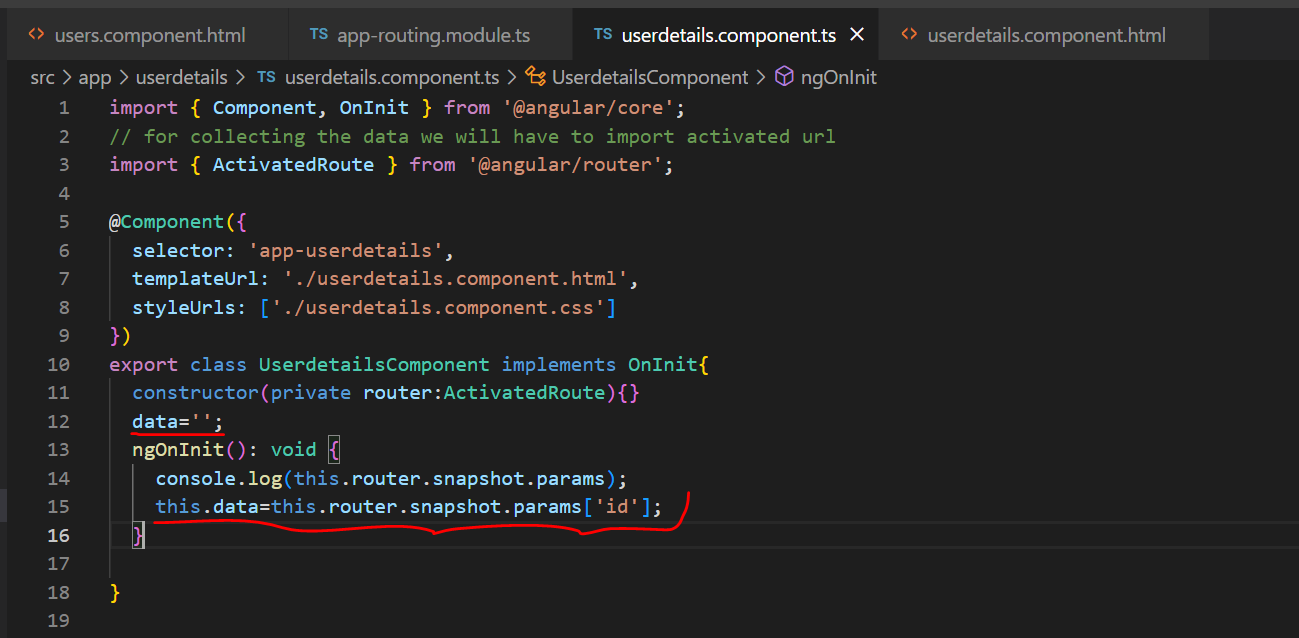


Output:

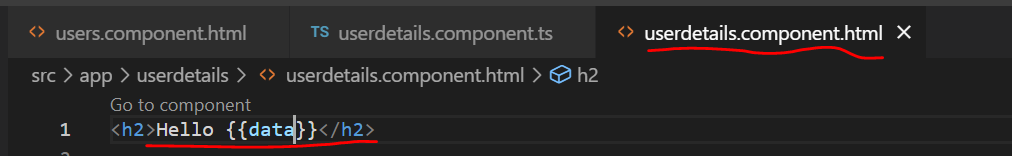


Let’s print the data from param to template.

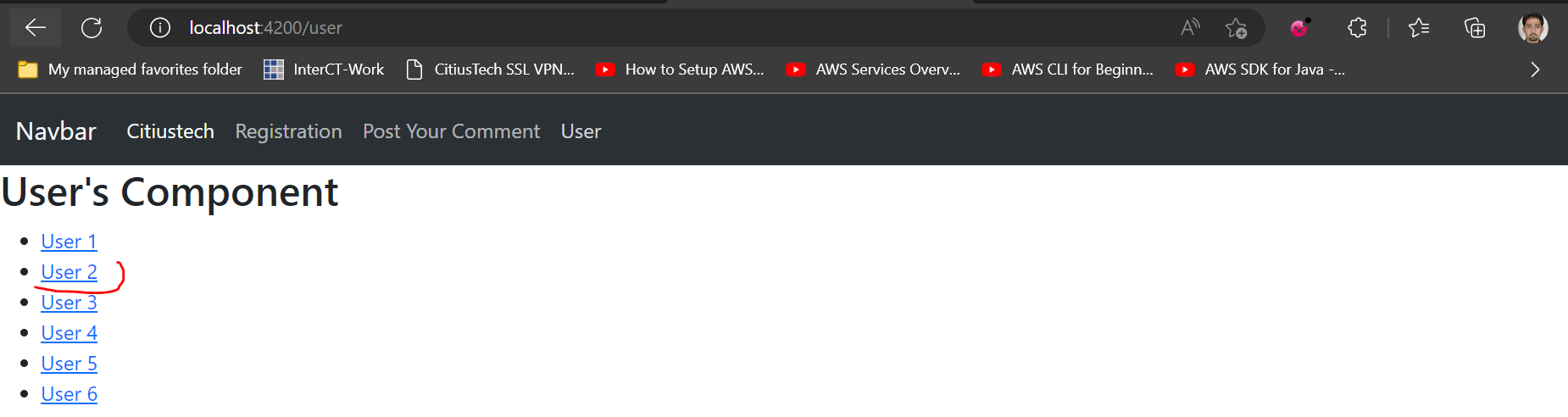
Open userdetail.component.ts and add given below code.



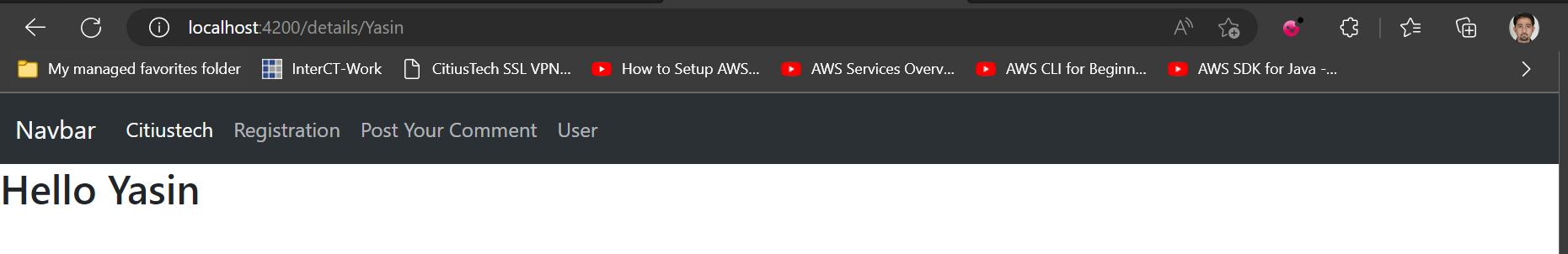
Open userdetails.component.html file and simply use interpolation.



Output:



Click on User2

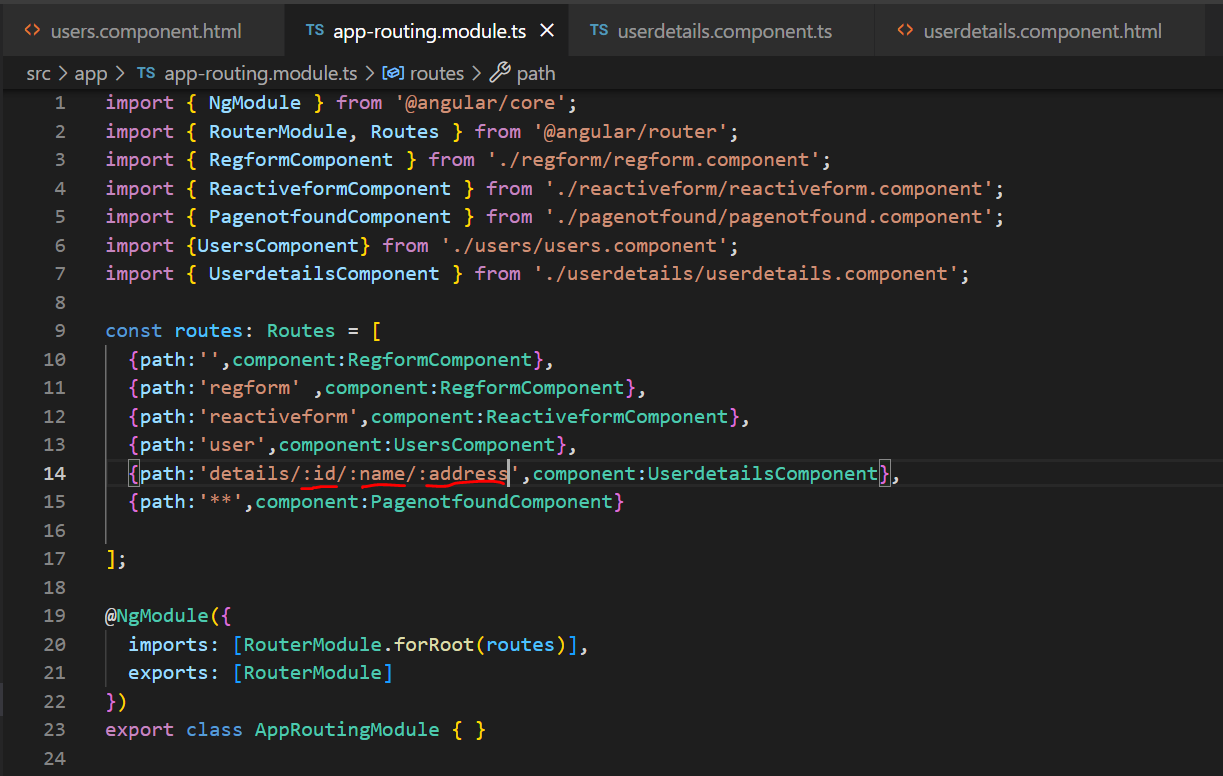


In this way, param can use with routing.

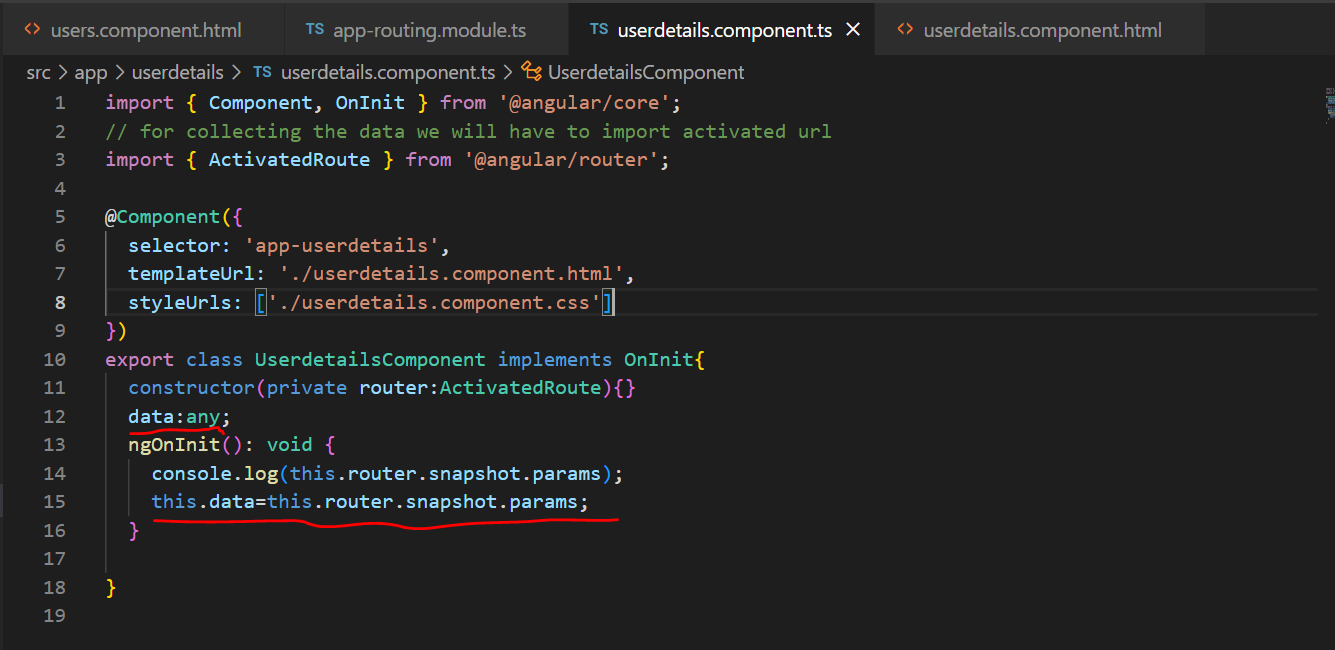
Let’s assume that we will have to display all details of any User then we can add multiple parameters too.

For achieving this we must do following changes.

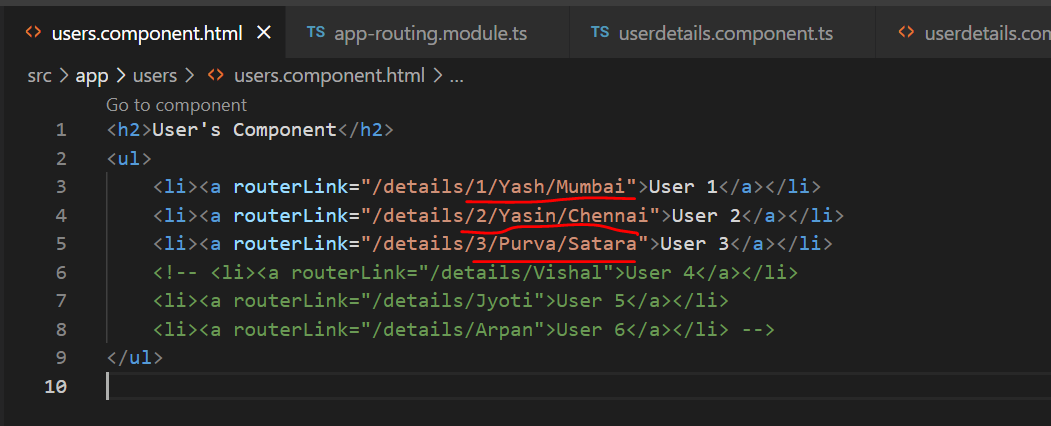
Open app-routing.module.ts



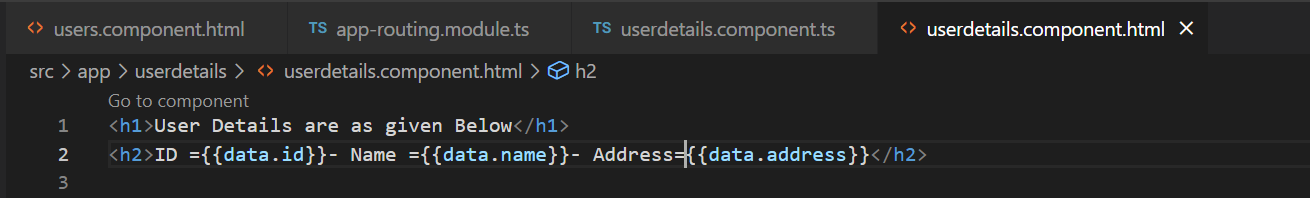
Open userdetail.component.t file and do changes



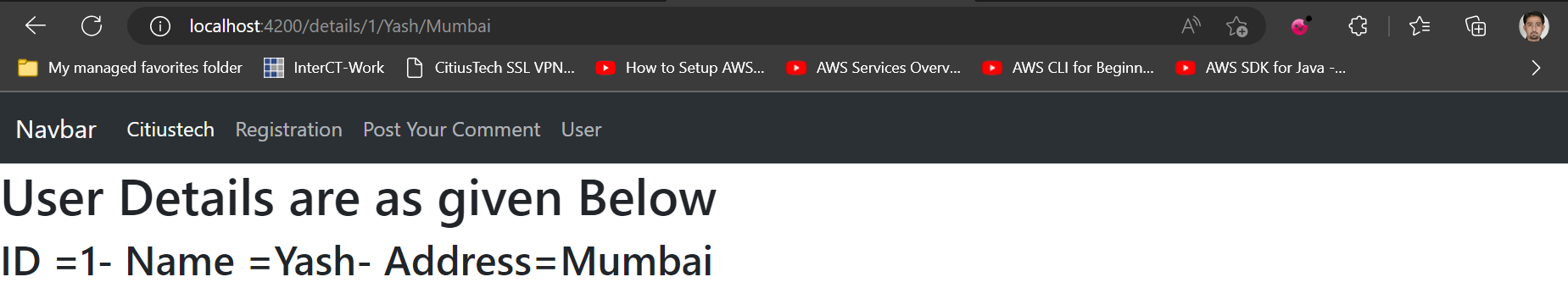
Open user.component.html and add three values in route



At Last using interpolation print data in userdetails.component.html



Output will look like as given below:



**Nested Routing:**

Nested Route is nothing but to route child routes.

Example, consider that we have User component already. So, we may have three children for user.

1. Prime user
2. Regular user
3. Guest.

So, from user component, if we will have to access child of user component then we can use this concept of nested routing.

It’s route will be something like

* http://localhost:4200/user/prime
* http://localhost:4200/user/regular
* <http://localhost:4200/user/guest>

Step by Step implementation of above demo.

**Step 1**. Open user.component.html. Open getBootstrap.com web site and we will copy readymade component.

We will use card. Can copy and paste given below code

Below code copied from getbootstrap.com and edited.

 <br/>

    <br/>

    <div class="container">

    <div class="card" style="width: 18rem;">

        <div class="card-body">

          <h5 class="card-title">Prime User</h5>

          <h6 class="card-subtitle mb-2 text-muted">Prime User Benefits</h6>

          <p class="card-text">Let's watch Newly launched movies on Prime Video</p>

          <a href="#" class="card-link">Know More</a>

        </div>

      </div>

      <div class="card" style="width: 18rem;">

        <div class="card-body">

          <h5 class="card-title">Regular User</h5>

          <h6 class="card-subtitle mb-2 text-muted">Regular User Benefits</h6>

          <p class="card-text">Let's watch some Free Old movies</p>

          <a href="#" class="card-link">Know More</a>

        </div>

      </div>

      <div class="card" style="width: 18rem;">

        <div class="card-body">

          <h5 class="card-title">Guest</h5>

          <h6 class="card-subtitle mb-2 text-muted">Guest</h6>

          <p class="card-text">Register First to get benefits</p>

          <a href="#" class="card-link">Know More</a>

        </div>

      </div>

    </div>

Css:

.container{

    text-align: center;

    display: flex;

}

.card{

    width: 40px;

    margin-right: 20px;

    margin-left: 50px;

}

Now, for our three types of users we can create we will have to create three nested components for User

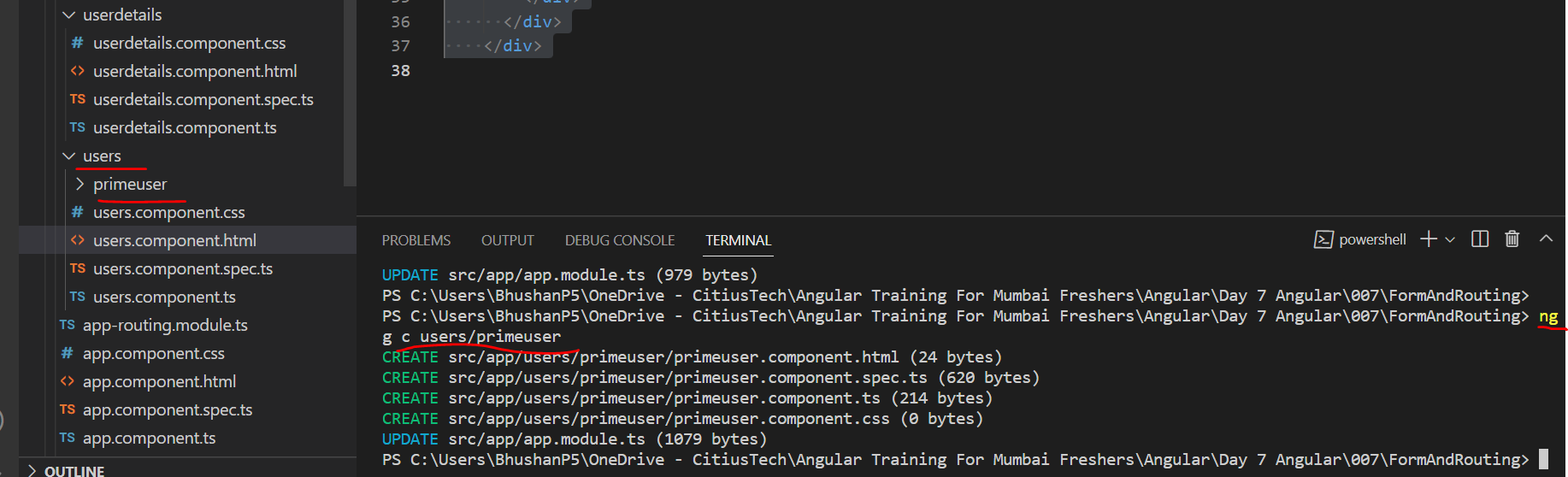
So our route will be like

<http://localhost:4200/User/prime>

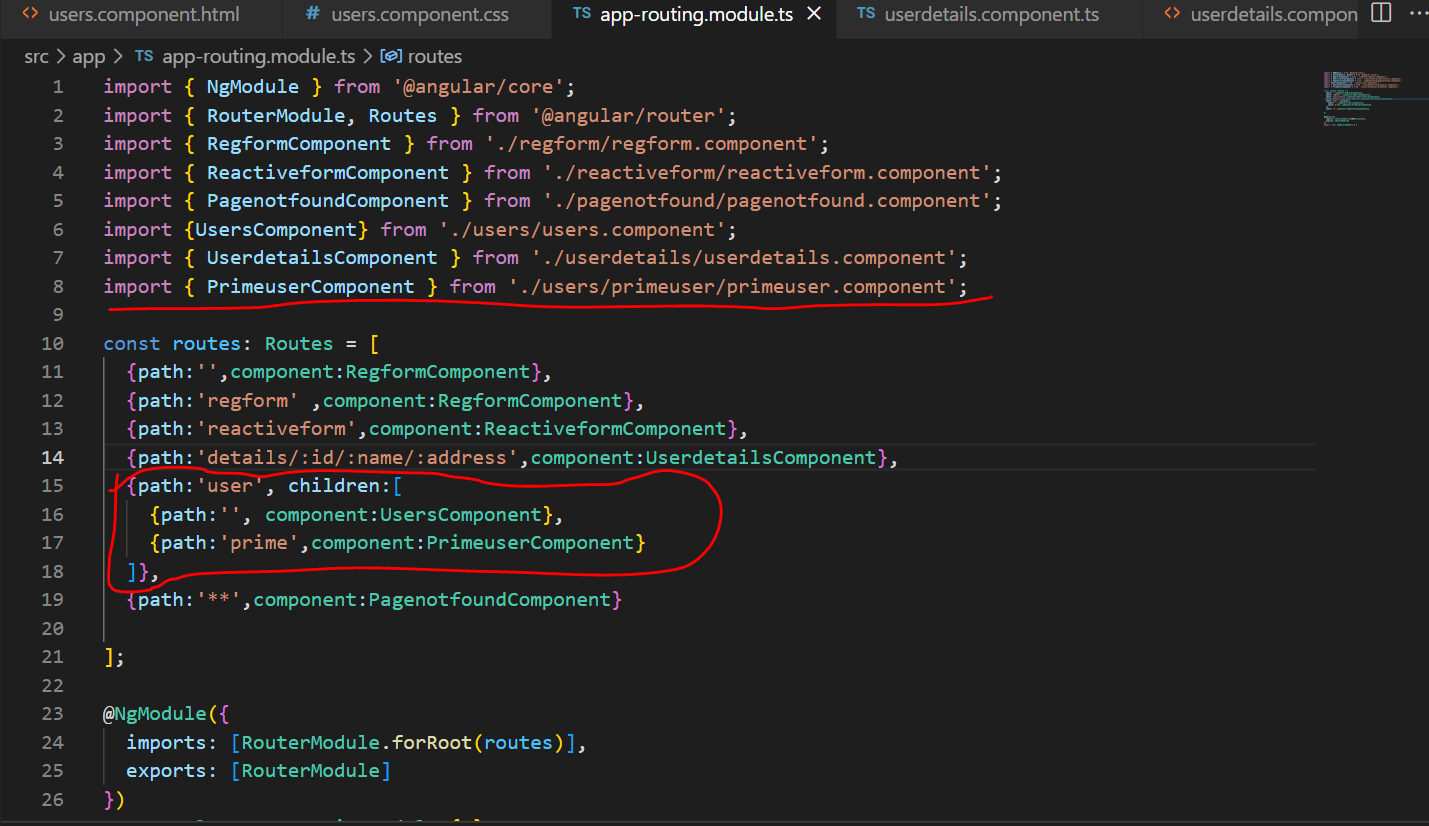
which will open prime component.

So, first we will add one component as prime component in product component.

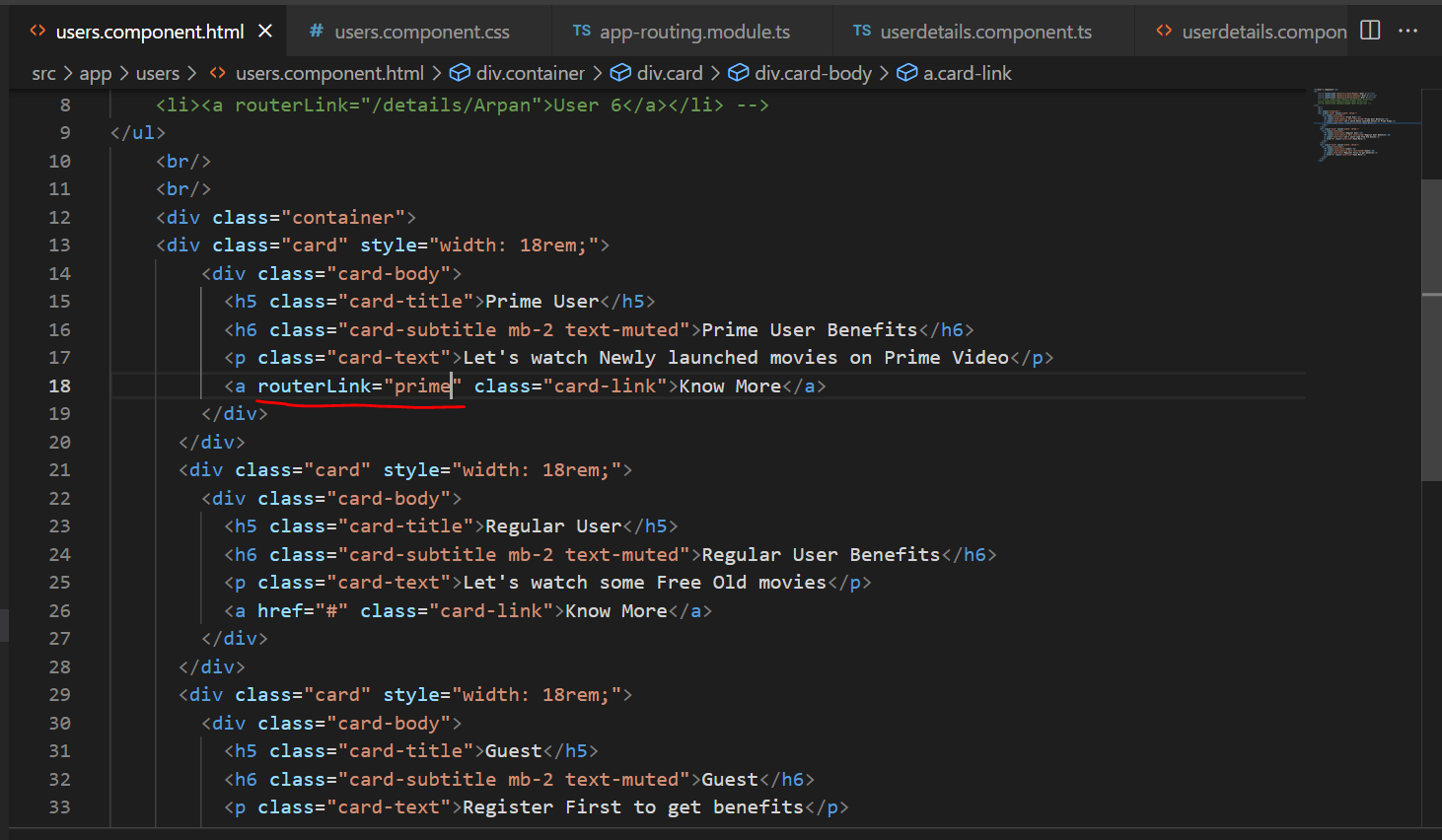
ng g c users/primeuser



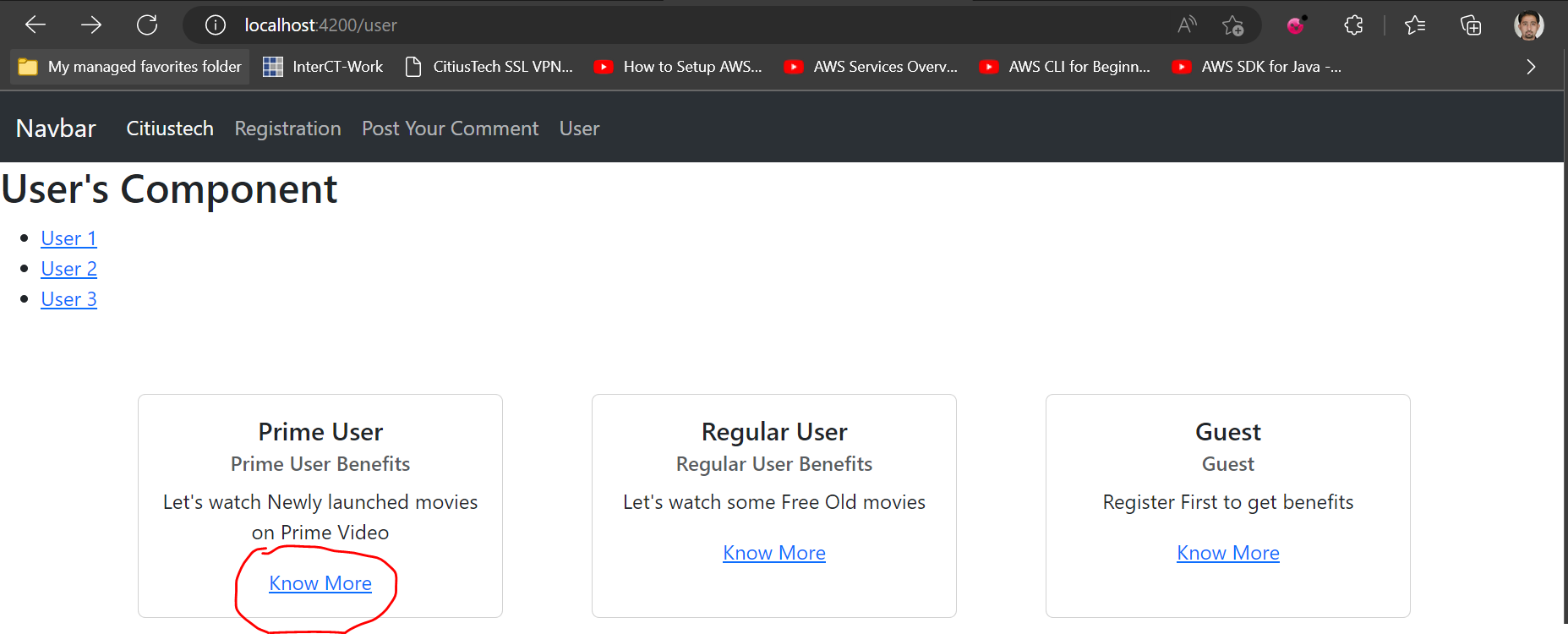
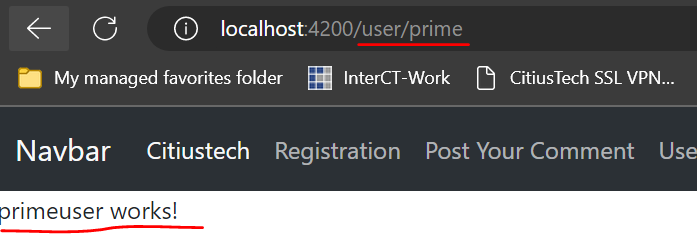
Open app-route.module.ts file and add given below code.



Now open user.component.html and add router link



Output:

**Angular Lazy Loading.**

**Lazy-loading feature modules**

By default, NgModules are eagerly loaded. This means that as soon as the application loads, so do all the NgModules, whether they are immediately necessary or not. For large applications with lots of routes, consider lazy loading —a design pattern that loads NgModules as needed. Lazy loading helps keep initial bundle sizes smaller, which in turn helps decrease load times.

So, here we will try to understand how we can achieve this.

We will follow given below steps.

1. Make a new module
2. Make two components
3. Implement lazy loading in routes.

**Step 1.** Create module along with routing module.

**Command:**

**ng g m lazymodule –routing**

It will create a module with two files.

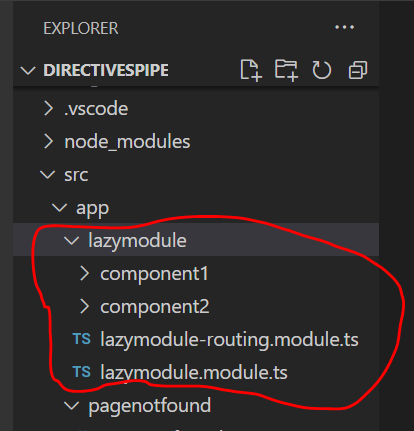


**Step 2.** Create two components in lazymodule

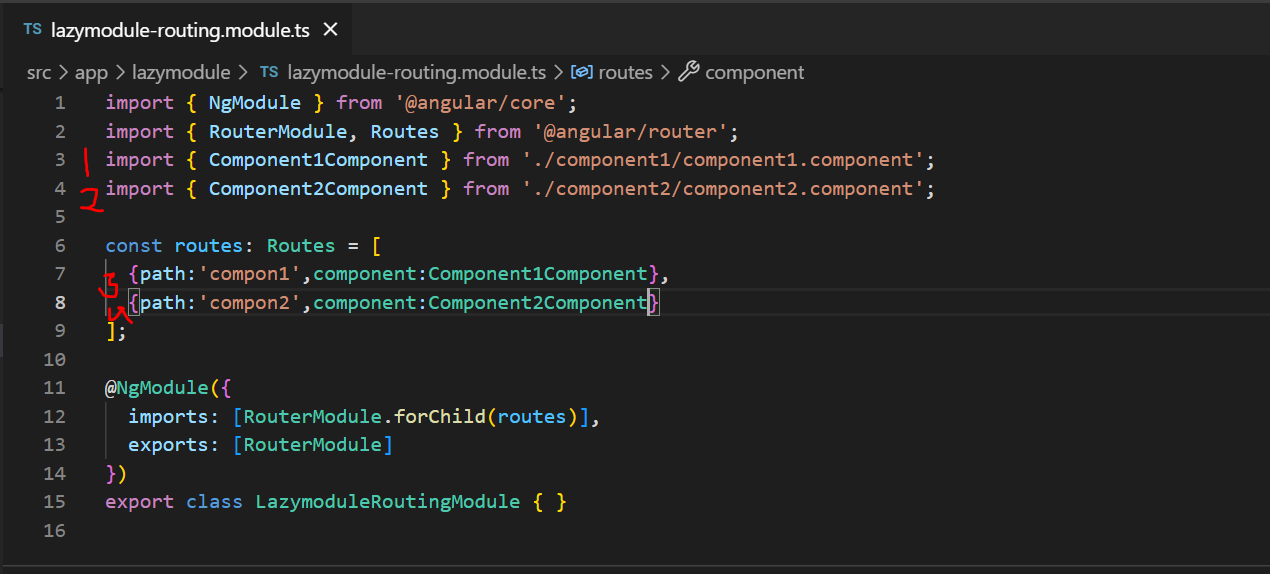
**Commands**

**ng g c lazymodule/component1**

**ng g c lazymodule/component2**



**Step 3.** Open **lazymodule-routing.module.ts.** Import both components in this class file on top.

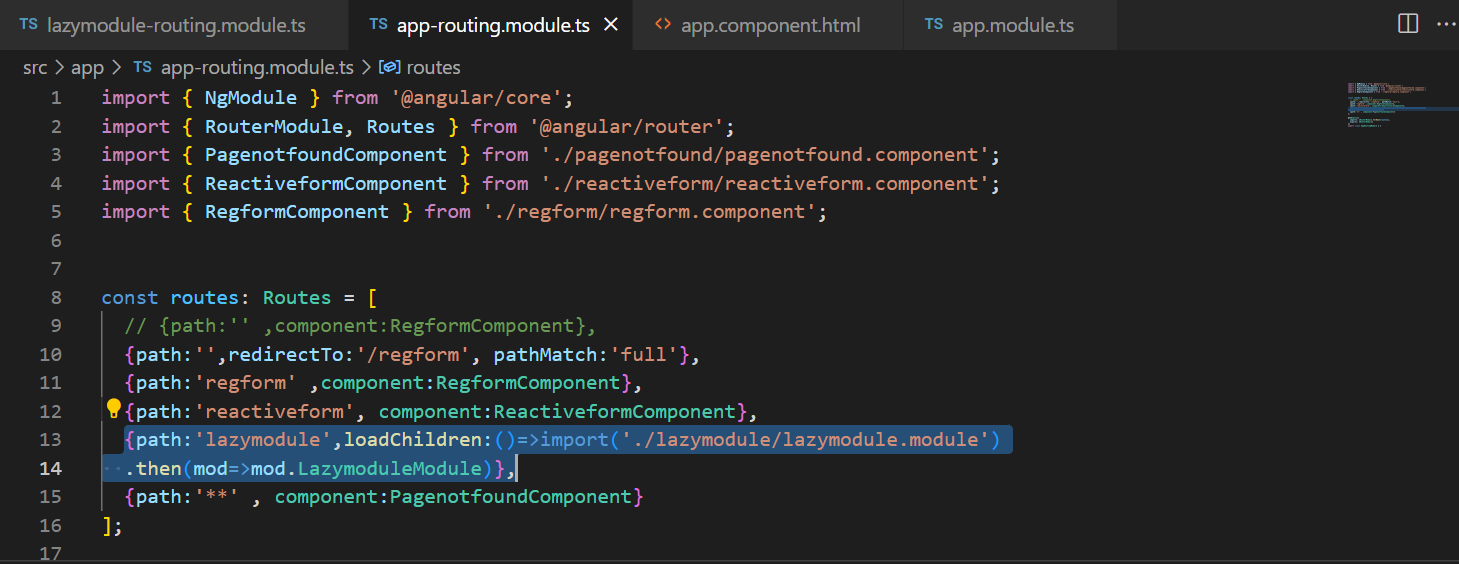


**Step 4.** Now, open **app-routing.module.ts** file

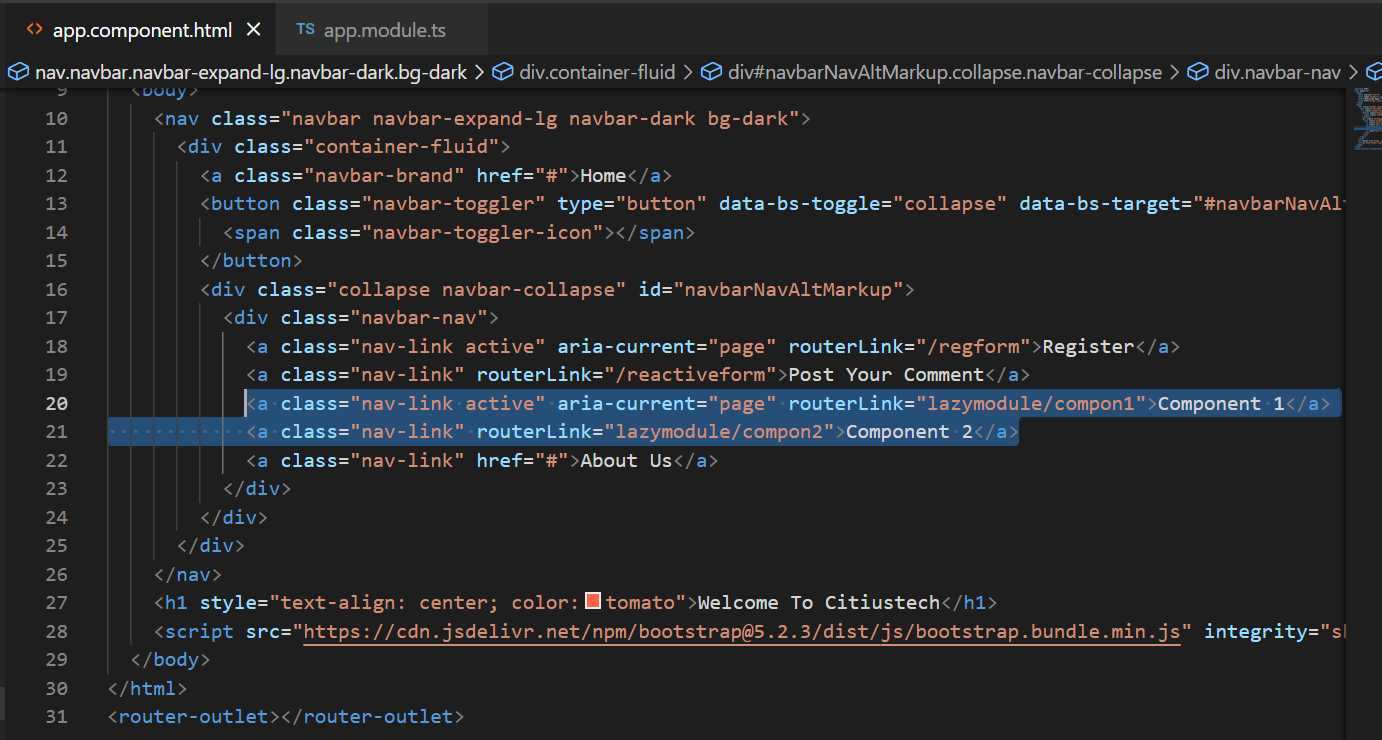
Add below code in **app-routing.module.ts.**

{path:'lazymodule',loadChildren:()=>import('./lazymodule/lazymodule.module')

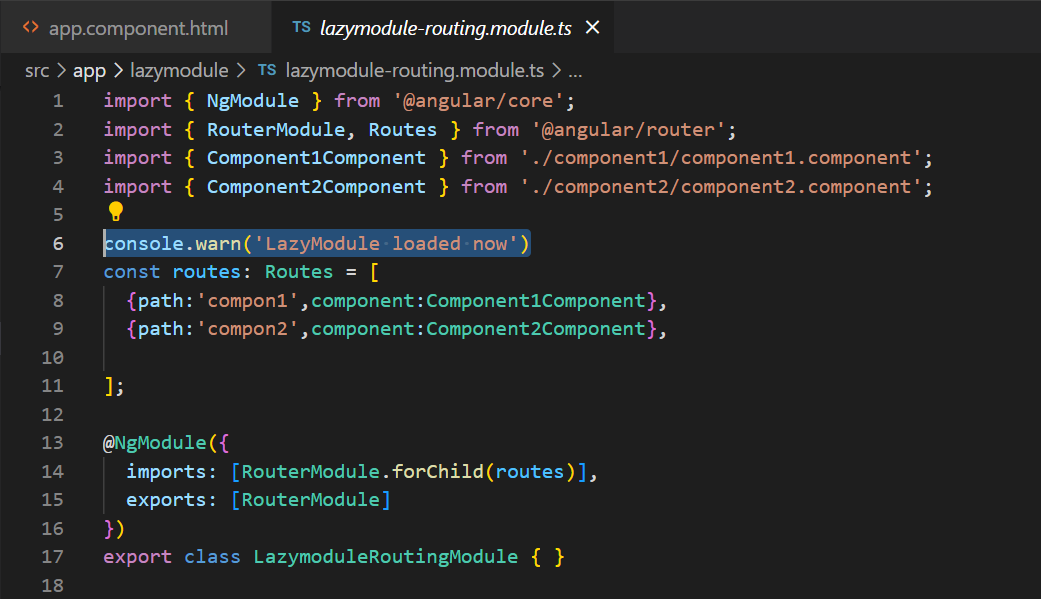
  .then(mod=>mod.LazymoduleModule)},



**Step 5.** Open **app.component.html**

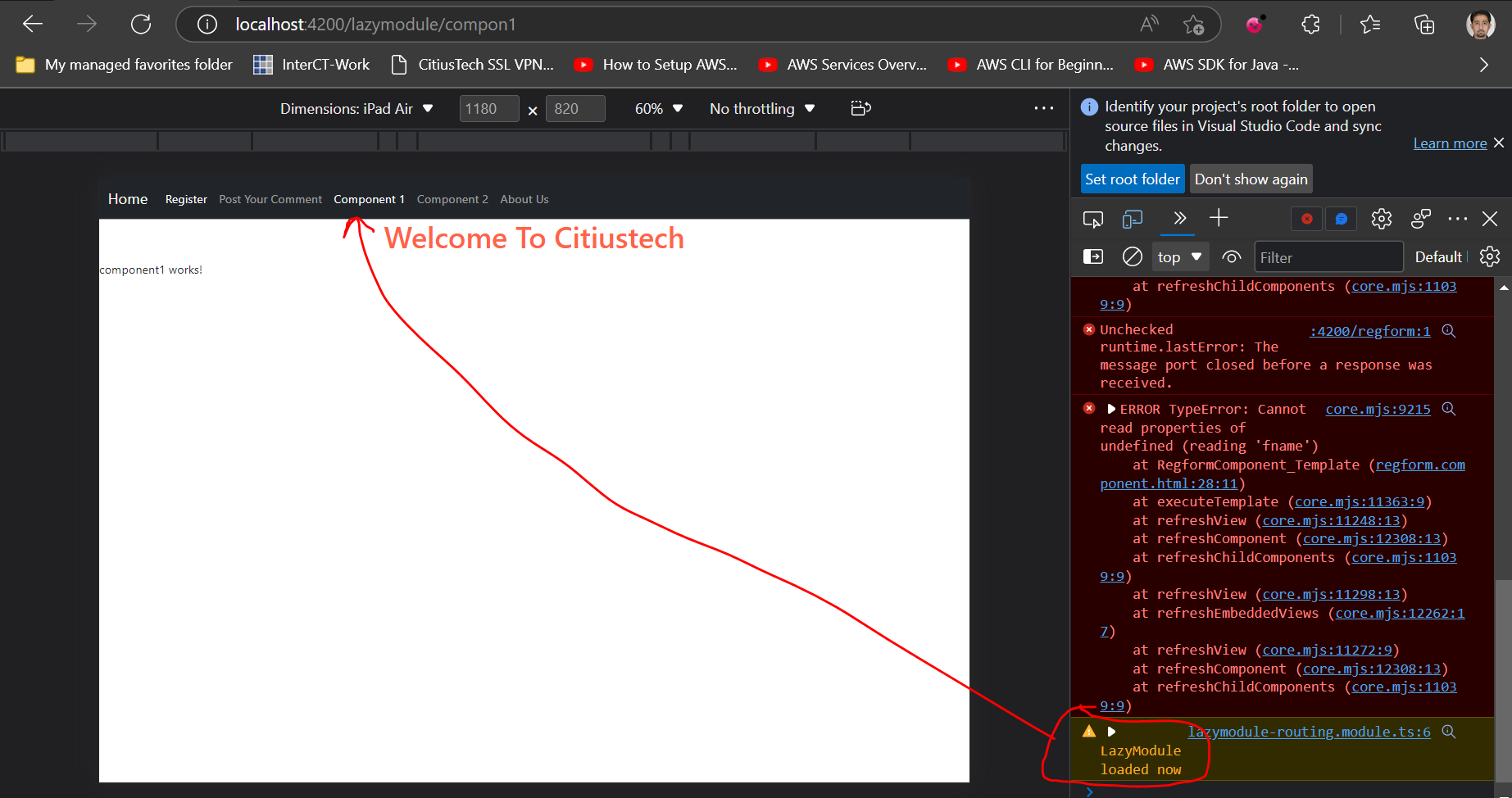


**Step 6.** In order to check whether it’s lazily loading or eagerly loading we will simply add one console in **lazymodule-routing.module.ts**



Now, open browser and inspect our page. Here you will get to know that these two components of lazy module won’t be loaded at first time when application is loading. By this way we can achieve lazy loading.

Output:



It will be loaded when we will click on component1.

**Communicating with external services (HTTP Communication using HttpClient)**

We have seen all core concepts of Angular and we know how we can get the data from service and display on templates.

But What if we will have to fetch data from remote resource and get it in our angular application.

So, here in such scenarios we will have to use HTTPClient to achieve this.

I hope everyone practiced services because here we will use service to communicate with remote service.

Let’s understand it with an example.

Here we required one API to communicate with. But for now we will use readymade fake API which is available that is json-placeholder.

**Step 1.** Here is url for our fake api

<https://jsonplaceholder.typicode.com/users>

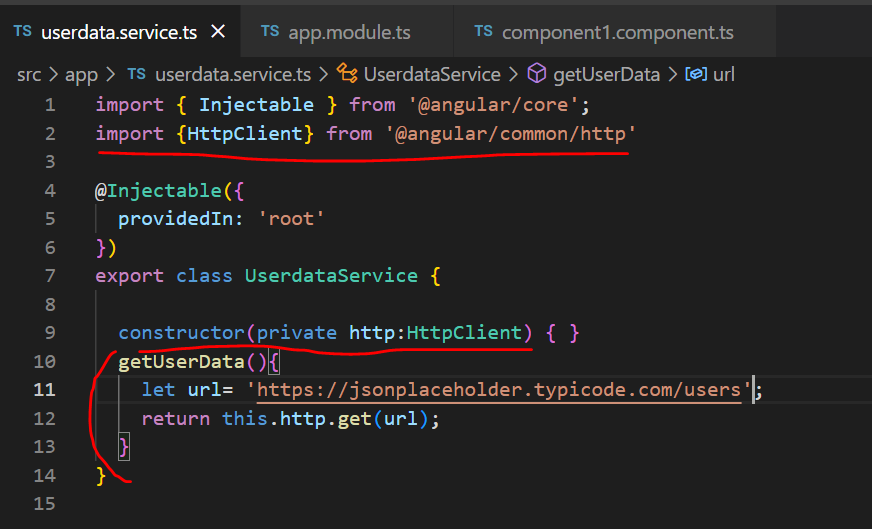
Our target is to connect with this API and get data from here.

**Step 2.** Create one service in our application.

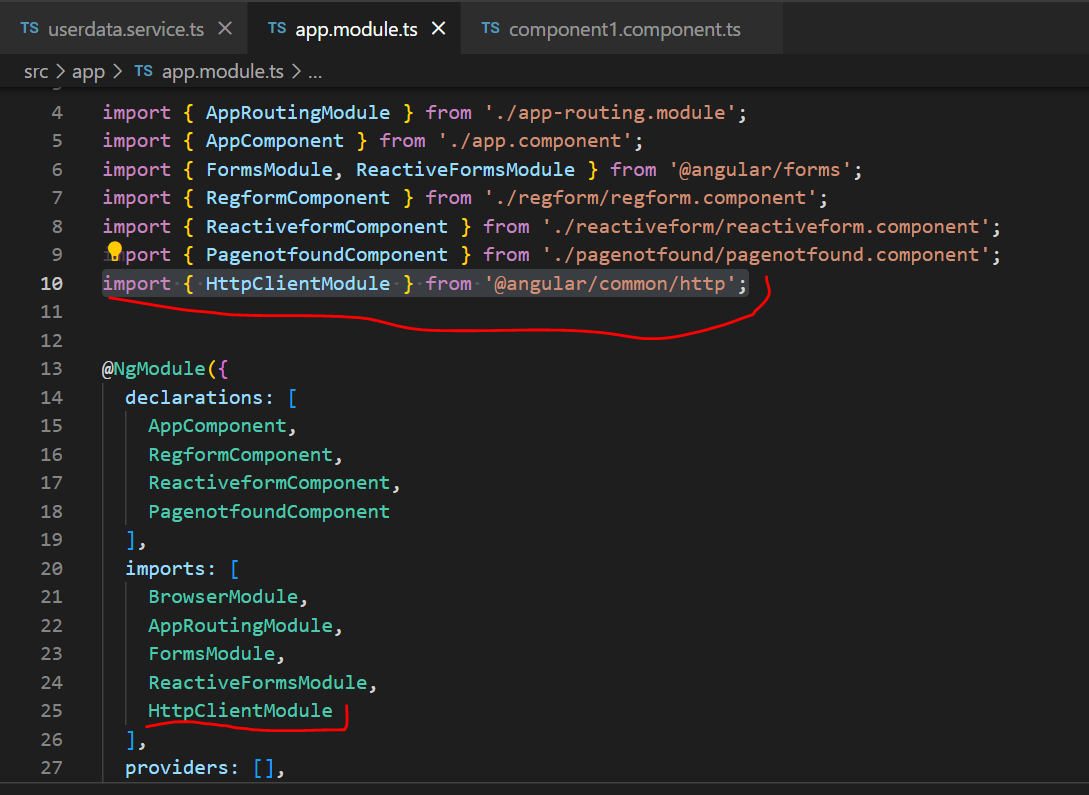
Command 🡪

**ng g s userdata**

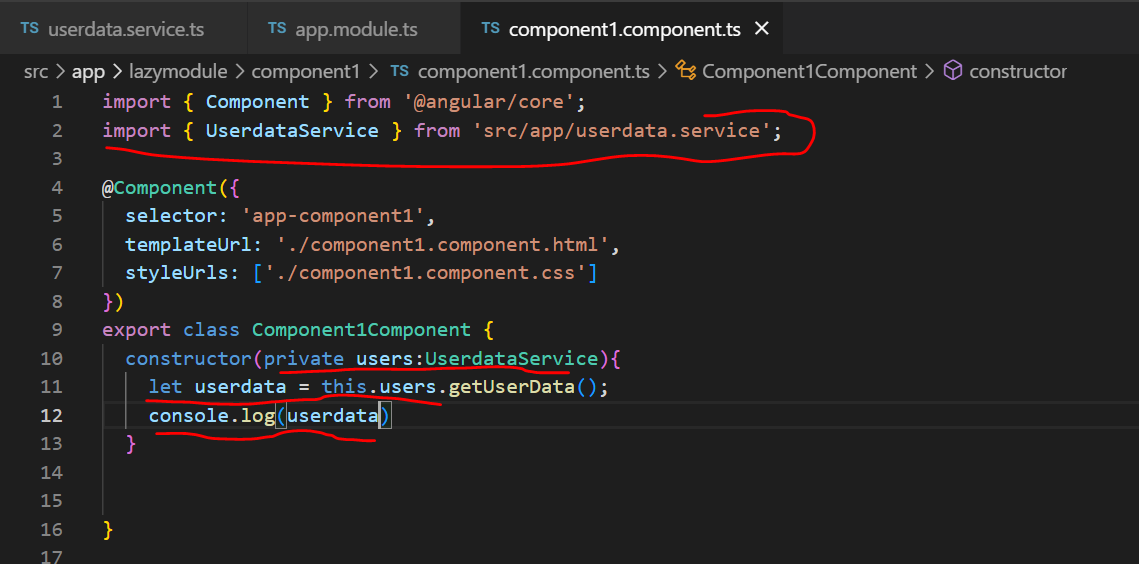
Open userdata.service.ts file and add given below code.



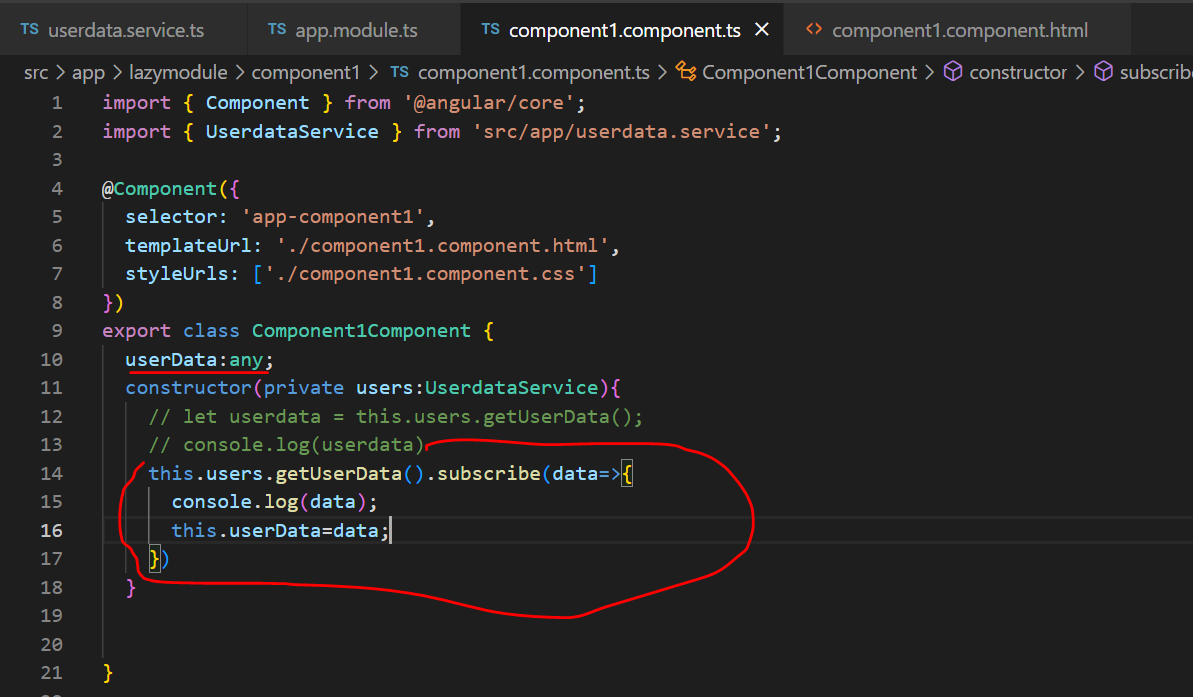
**Step 3:** Open **app.module.ts** file and import HttpClientModule.



**Step 4:** Let’s plan to display this data with our component1 of lazymodule component. So, open **component1.component.ts file.**

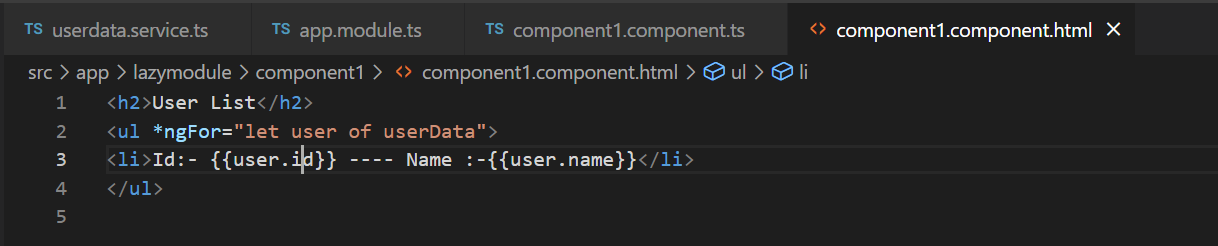


But in order to display data on template page.Write final version of **component1.component.ts file**



Display this userData on html page using interpolation

**Step 5.**  Add given below code in **component1.component.html**



Output :

