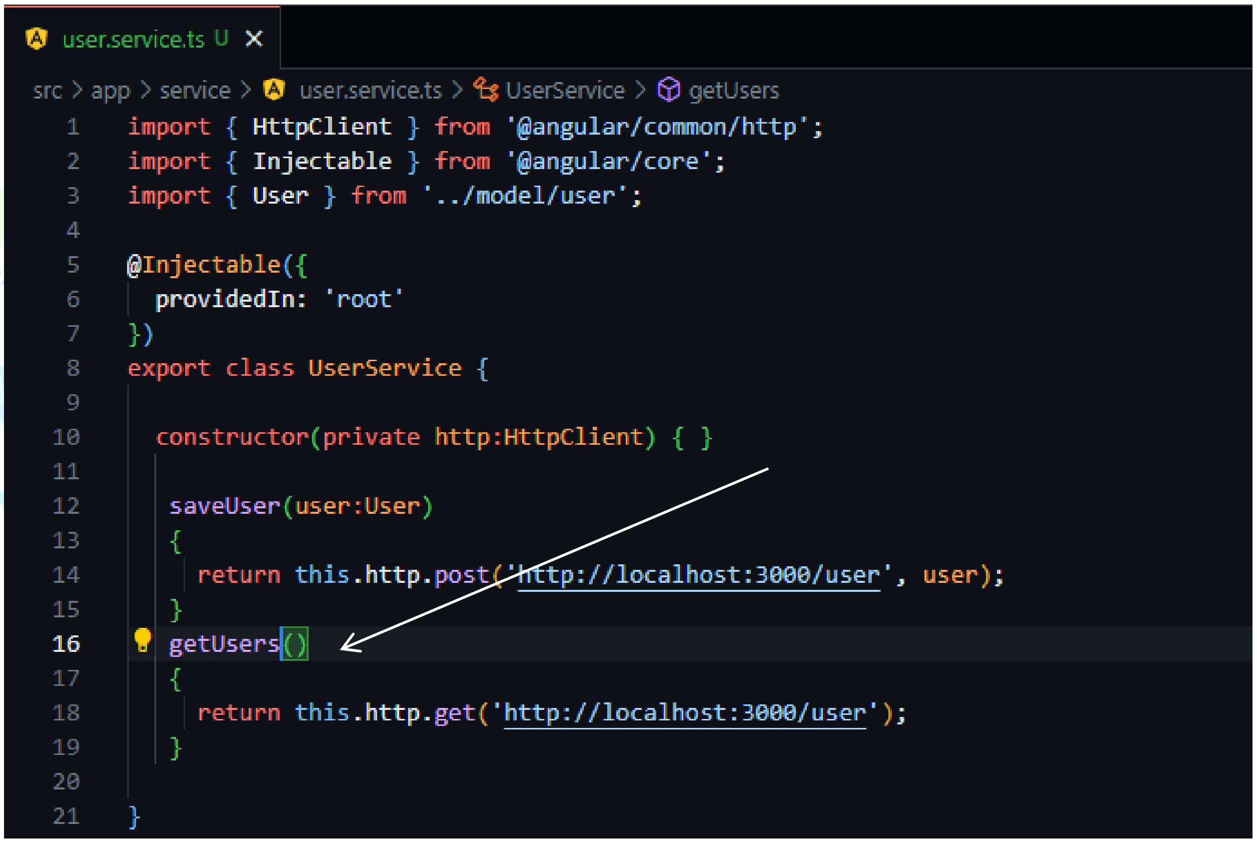
by Kunal Sir

C R U D G E T P U T D E L E T E

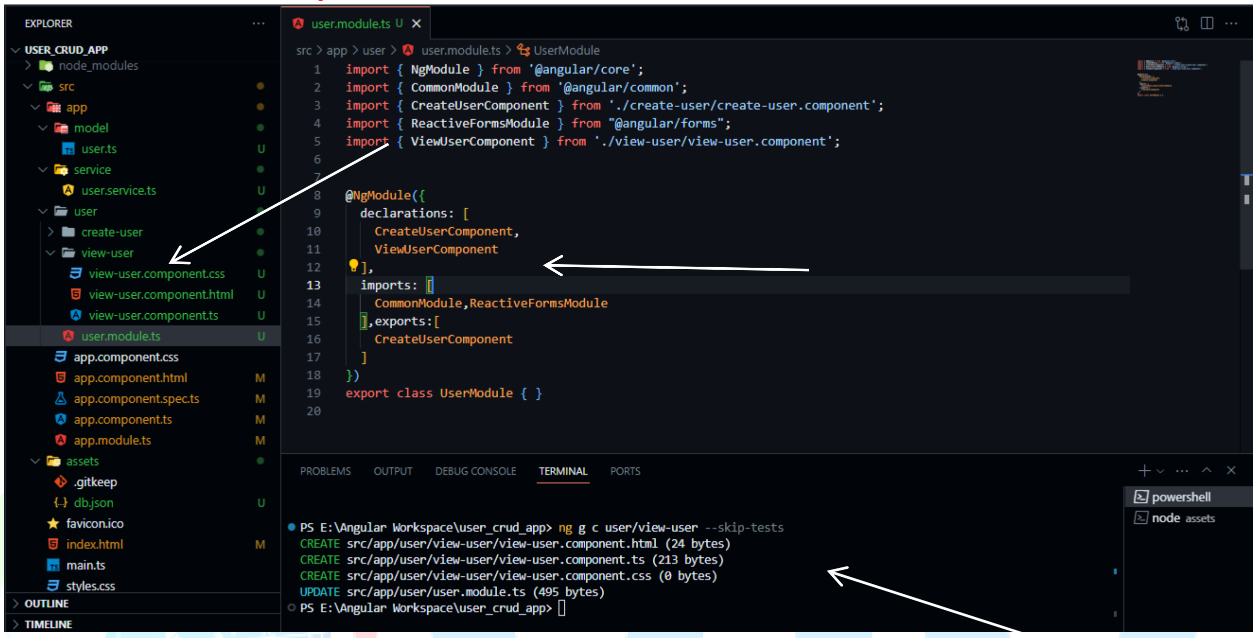
❖ Steps to consume data from Json-server mock service :-

1. **Declare getUsers function in UserService to consume user records using HttpClient's get() function.**


user.service.ts U ×
src > app > service > user.service.ts > UserService > getUsers

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { User } from '../model/user';

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

  constructor(private http:HttpClient) { }

   saveUser(user:User)
   {
     return this.http.post('http://localhost:3000/user', user);
   }
   getUsers()
   {
     return this.http.get('http://localhost:3000/user');
   }

}
```

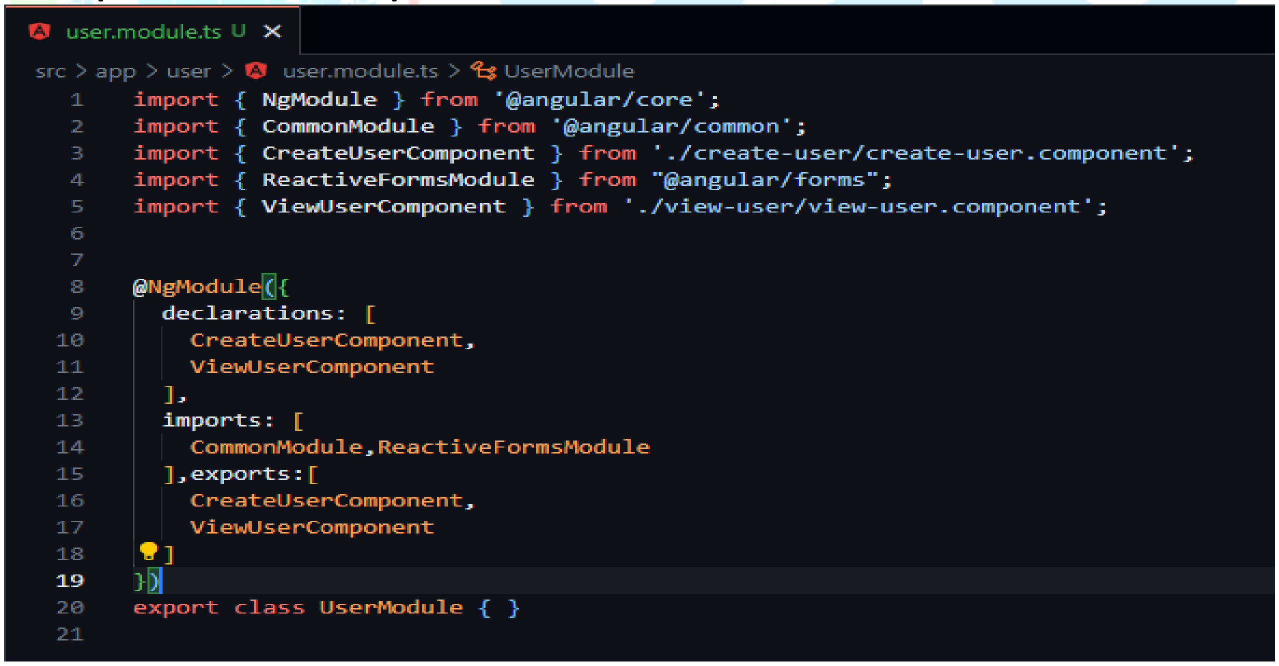
2. Create ViewUserComponent in UserModule using command : ng g c user/view-user --skip -tests



```typescript
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { CreateUserComponent } from './create-user/create-user.component';
import { ReactiveFormsModule } from "@angular/forms";
import { ViewUserComponent } from './view-user/view-user.component';


@NgModule({
  declarations: [
    CreateUserComponent,
    ViewUserComponent
  ],
  imports: [
    CommonModule,ReactiveFormsModule
  ],exports:[
    CreateUserComponent
  ]
})
export class UserModule { }
```

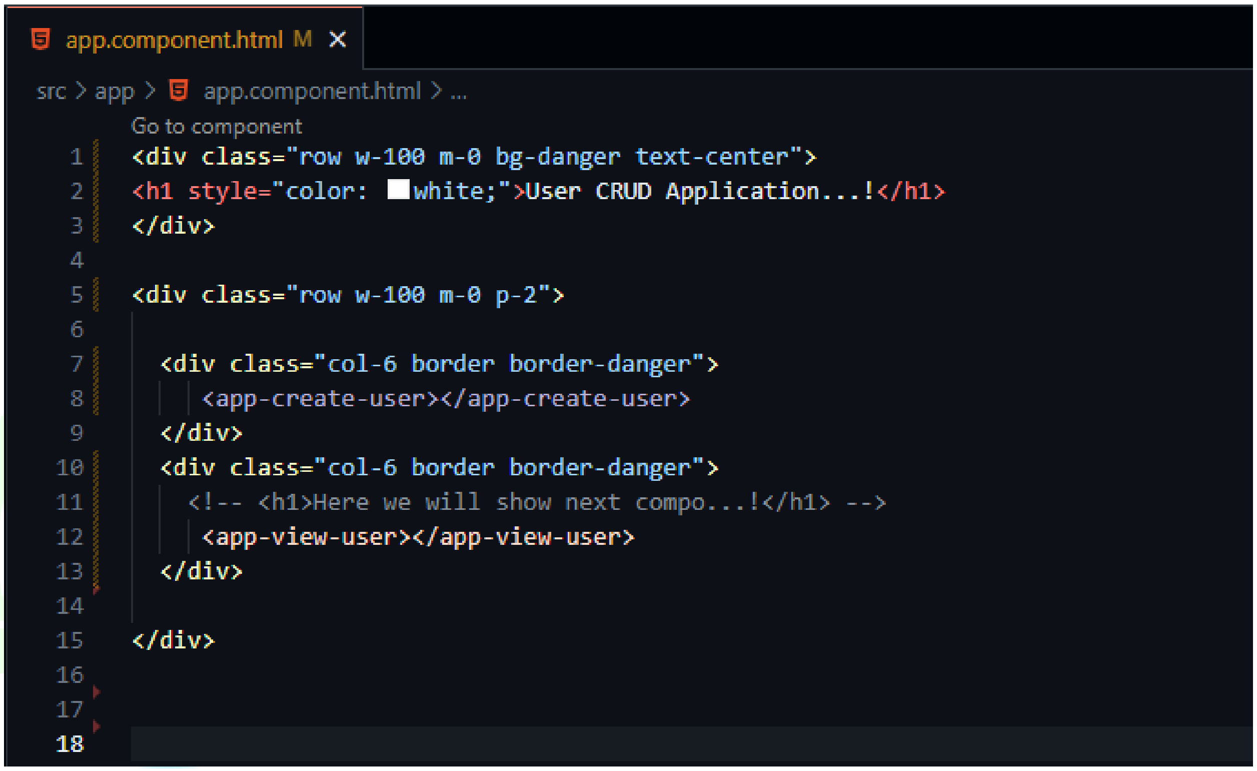
```
PS E:\Angular Workspace\user_crud_app> ng g c user/view-user --skip-tests
CREATE src/app/user/view-user/view-user.component.html (24 bytes)
CREATE src/app/user/view-user/view-user.component.ts (213 bytes)
CREATE src/app/user/view-user/view-user.component.css (0 bytes)
UPDATE src/app/user/user.module.ts (495 bytes)
PS E:\Angular Workspace\user_crud_app>
```

3. Export ViewUserComponent from UserModule.



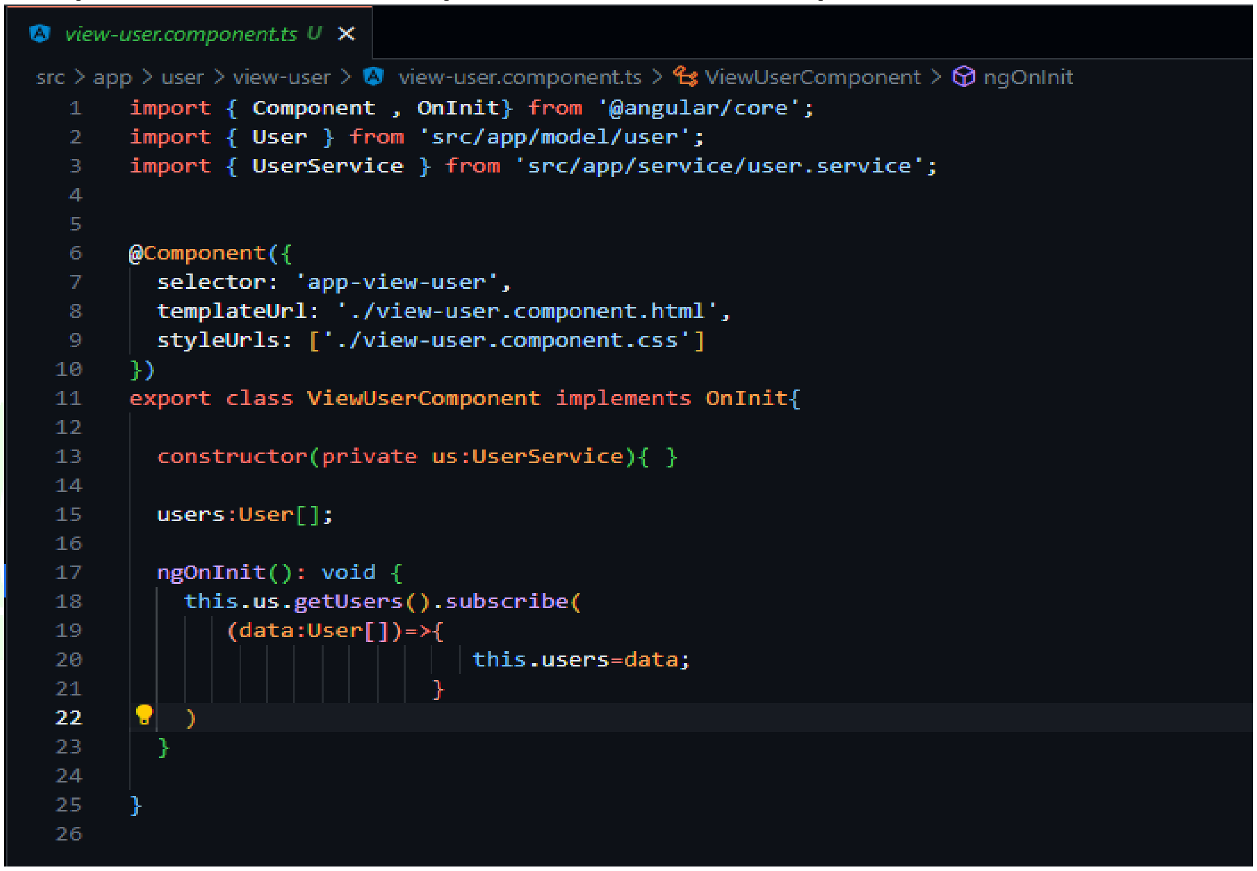
```typescript
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { CreateUserComponent } from './create-user/create-user.component';
import { ReactiveFormsModule } from "@angular/forms";
import { ViewUserComponent } from './view-user/view-user.component';


@NgModule({
  declarations: [
    CreateUserComponent,
    ViewUserComponent
  ],
  imports: [
    CommonModule,ReactiveFormsModule
  ],exports:[
    CreateUserComponent,
    ViewUserComponent
  ]
})
export class UserModule { }

```

4. **Add Selector of ViewUserComponent in app.component.html**



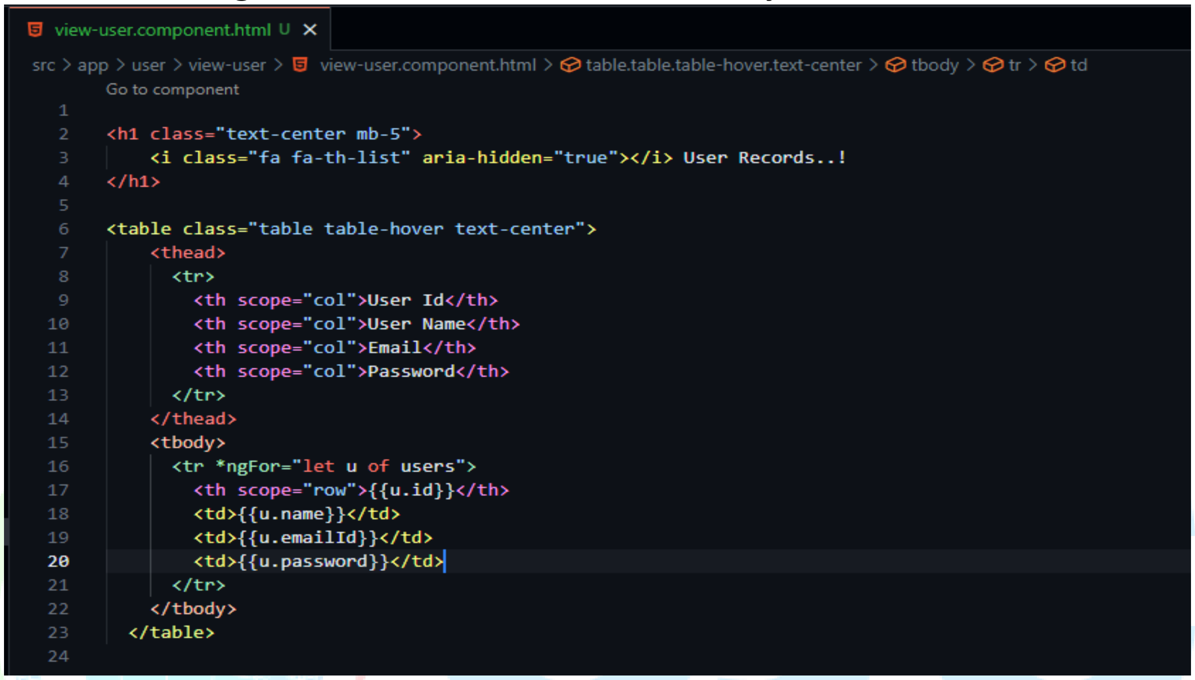
```
<div class="row w-100 m-0 bg-danger text-center">
<h1 style="color: white;">User CRUD Application...!</h1>
</div>

<div class="row w-100 m-0 p-2">

  <div class="col-6 border border-danger">
      <app-create-user></app-create-user>
  </div>
  <div class="col-6 border border-danger">
    <!-- <h1>Here we will show next compo...!</h1> -->
      <app-view-user></app-view-user>
  </div>

</div>
```

5. **In ViewUser component use ngOnInit hook to subscribe getUsers function and also store the data exposed by end-point into global property of component to access in template of a ViewUserComponent.**


view-user.component.ts U ×
src > app > user > view-user > view-user.component.ts > ViewUserComponent > ngOnInit

```
import { Component , OnInit} from '@angular/core';
import { User } from 'src/app/model/user';
import { UserService } from 'src/app/service/user.service';


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

  constructor(private us:UserService){ }

  users:User[];

  ngOnInit(): void {
    this.us.getUsers().subscribe(
        (data:User[])=>{
                            this.users=data;
                        }
    )
  }

}
```

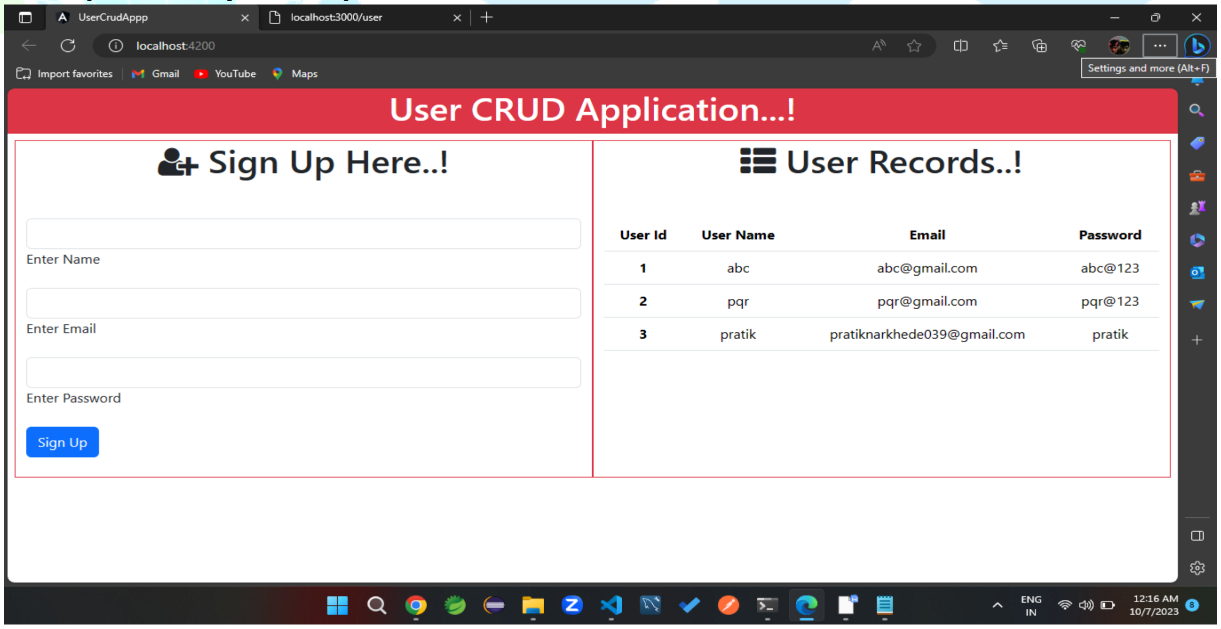
6. In view-user.component.html show users records of a users array in table format ,use *ngFor directive to iterate users array.



view-user.component.html U ×
src > app > user > view-user > view-user.component.html > table.table.table-hover.text-center > tbody > tr > td
Go to component

```html

<h1 class="text-center mb-5">
    <i class="fa fa-th-list" aria-hidden="true"></i> User Records..!
</h1>

<table class="table table-hover text-center">
    <thead>
      <tr>
        <th scope="col">User Id</th>
        <th scope="col">User Name</th>
        <th scope="col">Email</th>
        <th scope="col">Password</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let u of users">
        <th scope="row">{{u.id}}</th>
        <td>{{u.name}}</td>
        <td>{{u.emailId}}</td>
        <td>{{u.password}}</td>
      </tr>
    </tbody>
  </table>

```

➢ Upto now you output should look like as shown below



User CRUD Application...!

Sign Up Here..!

Enter Name

Enter Email

Enter Password

Sign Up

User Records..!

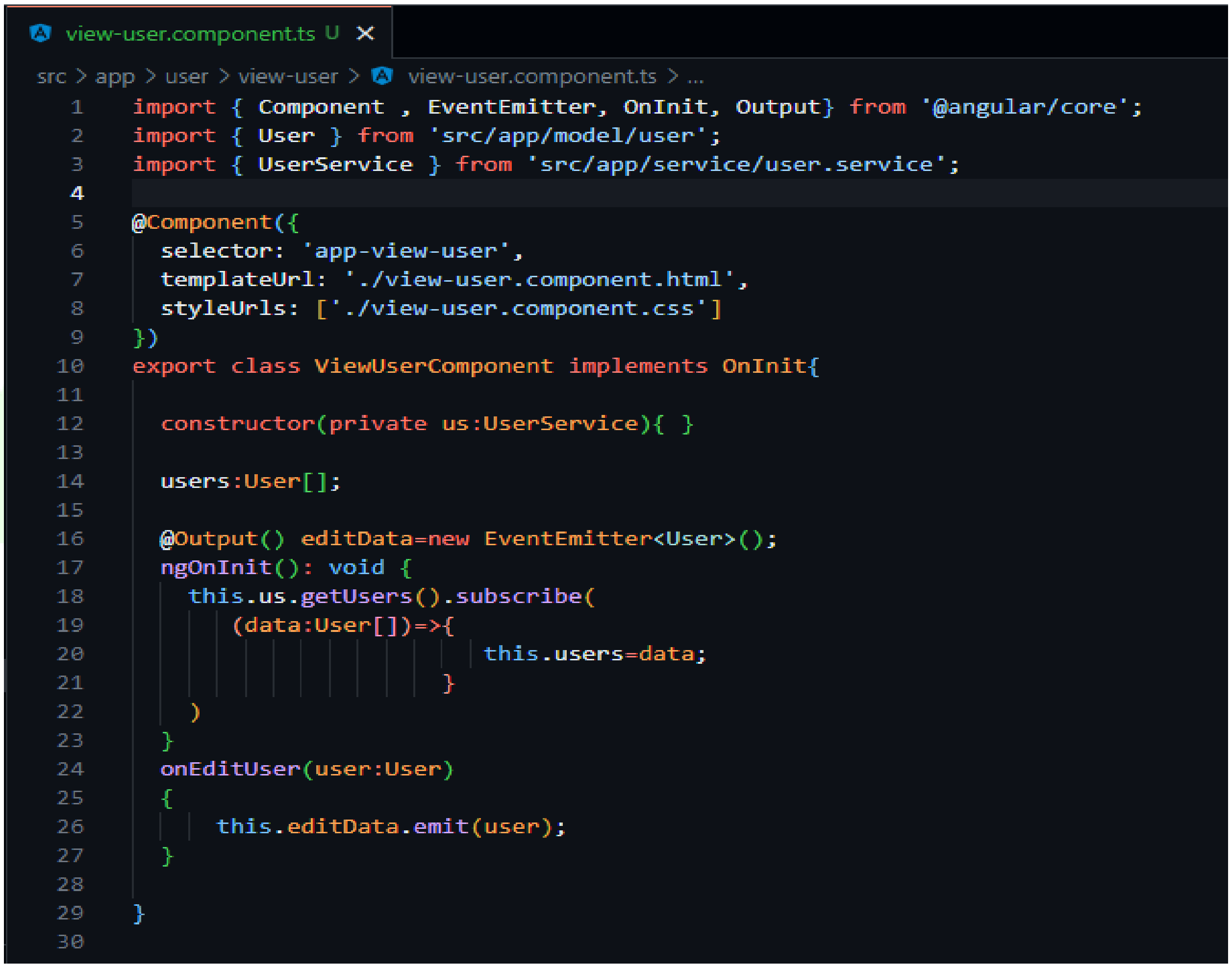
User Id	User Name	Email	Password
1	abc	abc@gmail.com	abc@123
2	pqr	pqr@gmail.com	pqr@123
3	pratik	pratiknarkhede039@gmail.com	pratik

by Kunal Sir

C R U D G E T P U T D E L E T E

❖ Steps to update existing user record

1. In ViewUserComponent declare one Output property using @Output type of EventEmitter and also have one onEditUser function , witch will accept user object as parameter argument.

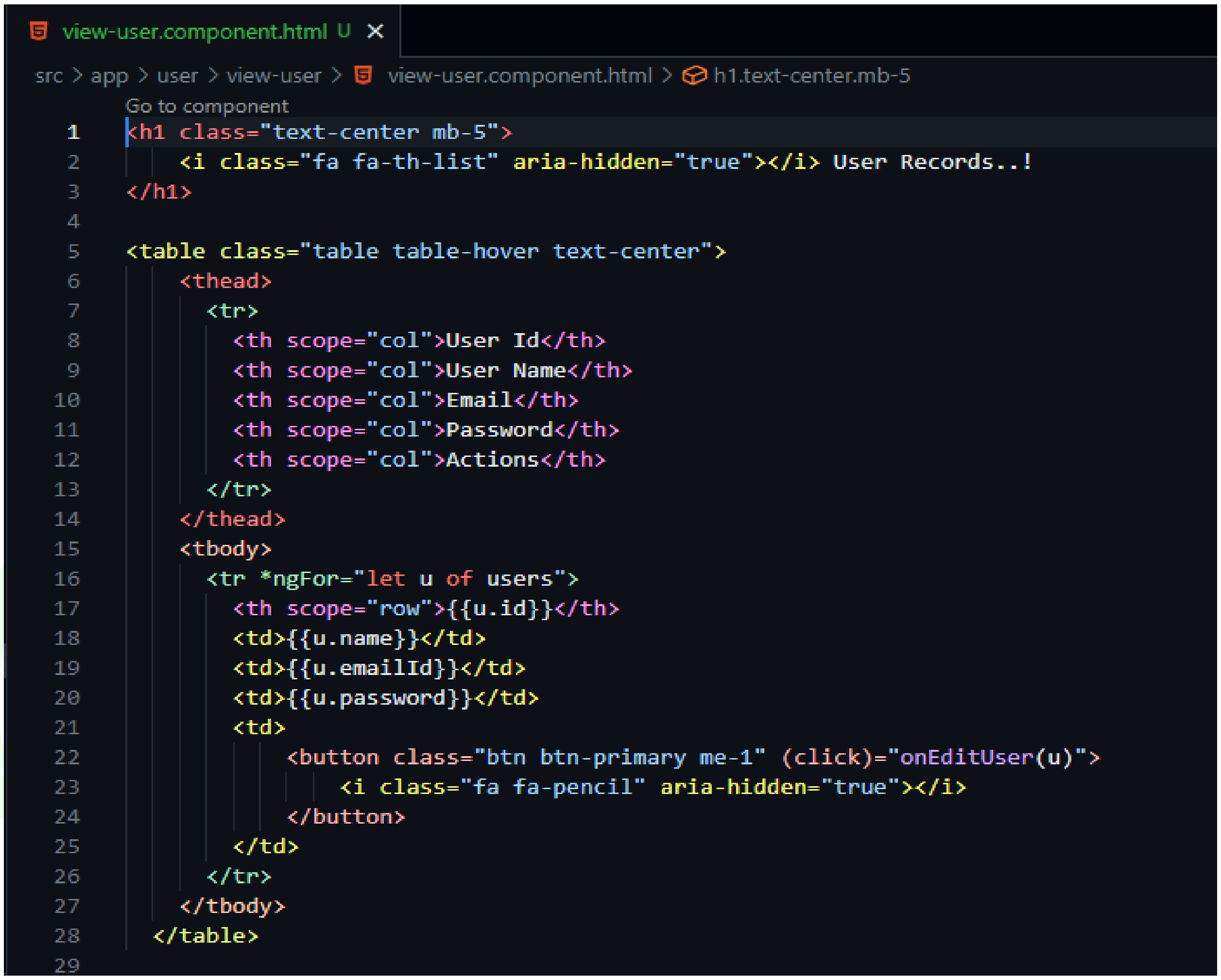


view-user.component.ts U ×

src > app > user > view-user > view-user.component.ts > ...

```
import { Component , EventEmitter, OnInit, Output} from '@angular/core';
import { User } from 'src/app/model/user';
import { UserService } from 'src/app/service/user.service';

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

  constructor(private us:UserService){ }

  users:User[];

  @Output() editData=new EventEmitter<User>();
  ngOnInit(): void {
    this.us.getUsers().subscribe(
        (data:User[])=>{
                    this.users=data;
                 }
    )
  }
  onEditUser(user:User)
  {
      this.editData.emit(user);
  }

}
```

2. In view-user.component.html add one button and on click event of that button bind the onEditUser function and pass user object as parameter argument as shown below.



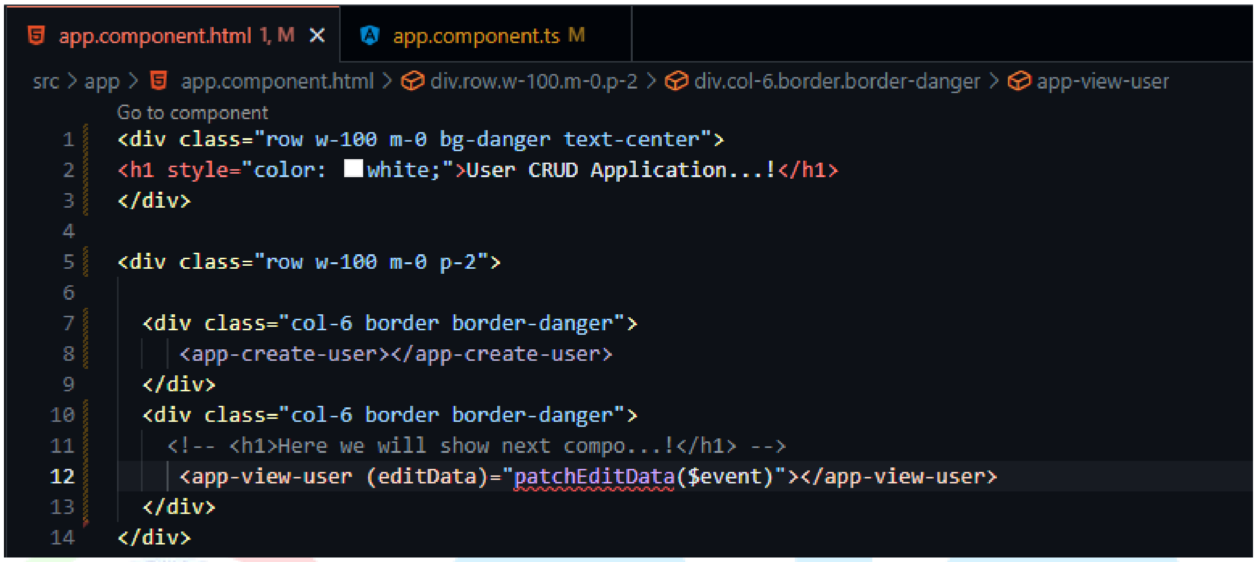
view-user.component.html U ×

src > app > user > view-user > view-user.component.html > h1.text-center.mb-5

Go to component

```html
<h1 class="text-center mb-5">
    <i class="fa fa-th-list" aria-hidden="true"></i> User Records..!
</h1>

<table class="table table-hover text-center">
    <thead>
      <tr>
        <th scope="col">User Id</th>
        <th scope="col">User Name</th>
        <th scope="col">Email</th>
        <th scope="col">Password</th>
        <th scope="col">Actions</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let u of users">
        <th scope="row">{{u.id}}</th>
        <td>{{u.name}}</td>
        <td>{{u.emailId}}</td>
        <td>{{u.password}}</td>
        <td>
            <button class="btn btn-primary me-1" (click)="onEditUser(u)">
                <i class="fa fa-pencil" aria-hidden="true"></i>
            </button>
        </td>
      </tr>
    </tbody>
  </table>
```

3. **In app.component.html bind the editdata property of ViewUserComponent using event binding as it will emit the data on event as shown below.**



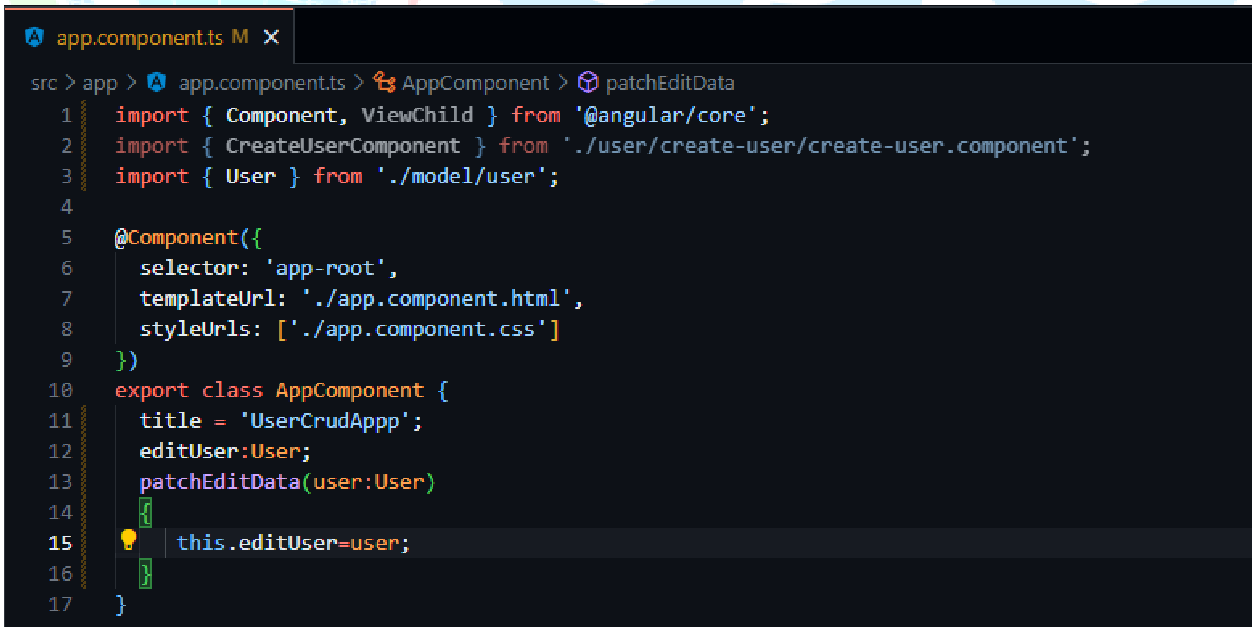
app.component.html 1, M × | app.component.ts M

src > app > app.component.html > div.row.w-100.m-0.p-2 > div.col-6.border.border-danger > app-view-user

Go to component

```html
<div class="row w-100 m-0 bg-danger text-center">
<h1 style="color: white;">User CRUD Application...!</h1>
</div>

<div class="row w-100 m-0 p-2">

  <div class="col-6 border border-danger">
     <app-create-user></app-create-user>
  </div>
  <div class="col-6 border border-danger">
    <!-- <h1>Here we will show next compo...!</h1> -->
     <app-view-user (editData)="patchEditData($event)"></app-view-user>
  </div>
</div>
```

4. Now declare patchEditData function which will accept user object as argument , also assign that object to global property in AppComponent.

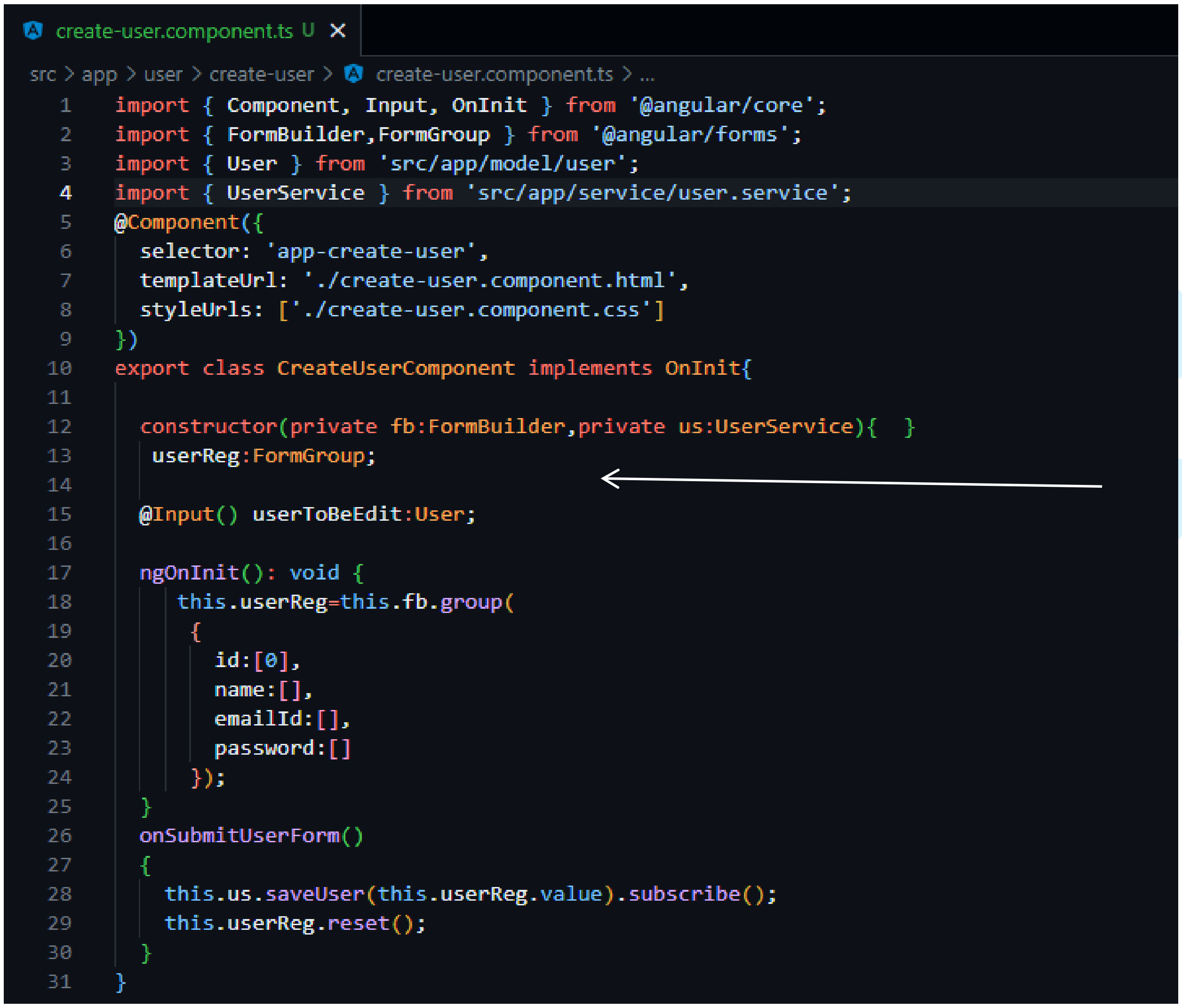


app.component.ts M ×

src > app > app.component.ts > AppComponent > patchEditData

```typescript
import { Component, ViewChild } from '@angular/core';
import { CreateUserComponent } from './user/create-user/create-user.component';
import { User } from './model/user';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'UserCrudAppp';
  editUser:User;
  patchEditData(user:User)
  {
     this.editUser=user;
  }
}
```

5. In CreateUserComponent declare one input property using @input decorator to receive
User details to be edited from parent component.

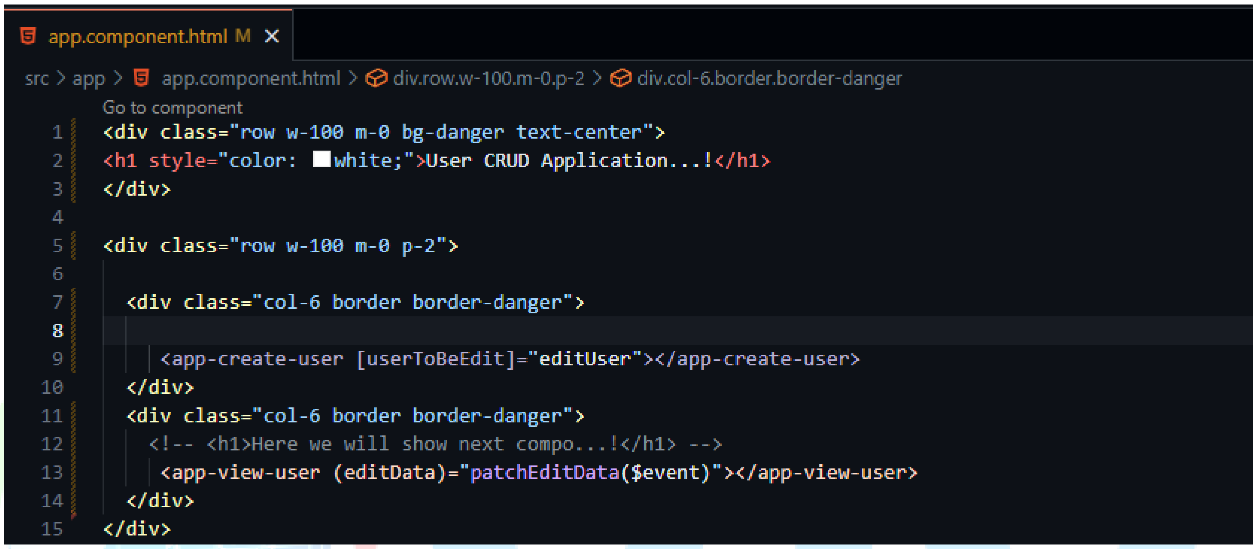


create-user.component.ts U

src > app > user > create-user > create-user.component.ts > ...

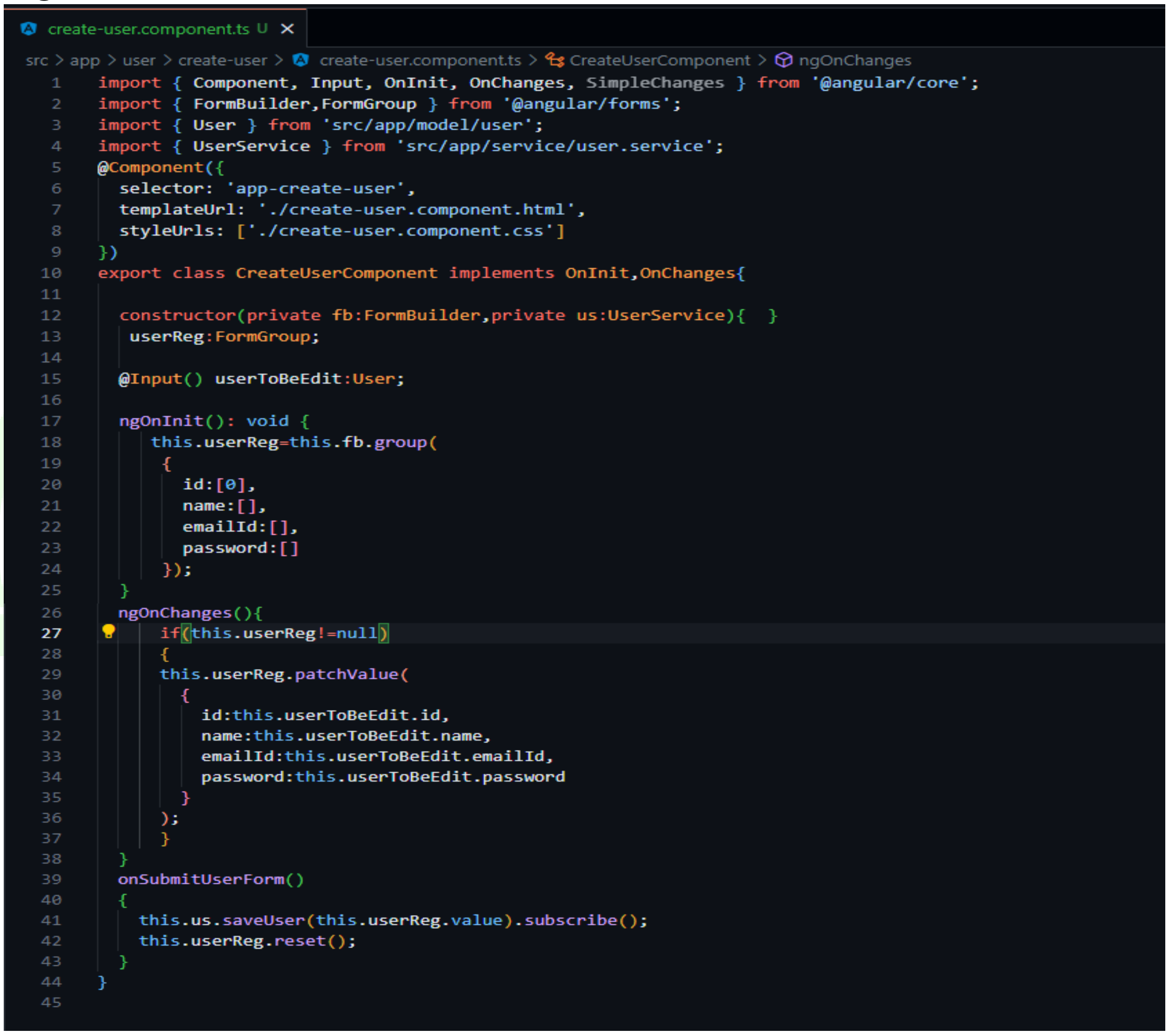
```
import { Component, Input, OnInit } from '@angular/core';
import { FormBuilder,FormGroup } from '@angular/forms';
import { User } from 'src/app/model/user';
import { UserService } from 'src/app/service/user.service';
@Component({
  selector: 'app-create-user',
  templateUrl: './create-user.component.html',
  styleUrls: ['./create-user.component.css']
})
export class CreateUserComponent implements OnInit{

  constructor(private fb:FormBuilder,private us:UserService){  }
   userReg:FormGroup;

  @Input() userToBeEdit:User;

  ngOnInit(): void {
     this.userReg=this.fb.group(
      {
        id:[0],
        name:[],
        emailId:[],
        password:[]
      });
  }
  onSubmitUserForm()
  {
    this.us.saveUser(this.userReg.value).subscribe();
    this.userReg.reset();
  }
}
```

6. **Now bind this userToBeEdit property in app.component.html with proerty editUser as shown below. To transfer user details from AppComponent to CreateUserComponent.**


app.component.html M ×
src > app > app.component.html > div.row.w-100.m-0.p-2 > div.col-6.border.border-danger
Go to component

```html
<div class="row w-100 m-0 bg-danger text-center">
<h1 style="color: white;">User CRUD Application...!</h1>
</div>

<div class="row w-100 m-0 p-2">

  <div class="col-6 border border-danger">

     <app-create-user [userToBeEdit]="editUser"></app-create-user>
  </div>
  <div class="col-6 border border-danger">
    <!-- <h1>Here we will show next compo...!</h1> -->
     <app-view-user (editData)="patchEditData($event)"></app-view-user>
  </div>
</div>
```

7. **Again in CreateUserComponent use ngOnChanges hoot to monitor the changes in input property and patch the value of input property to user-registration form.**

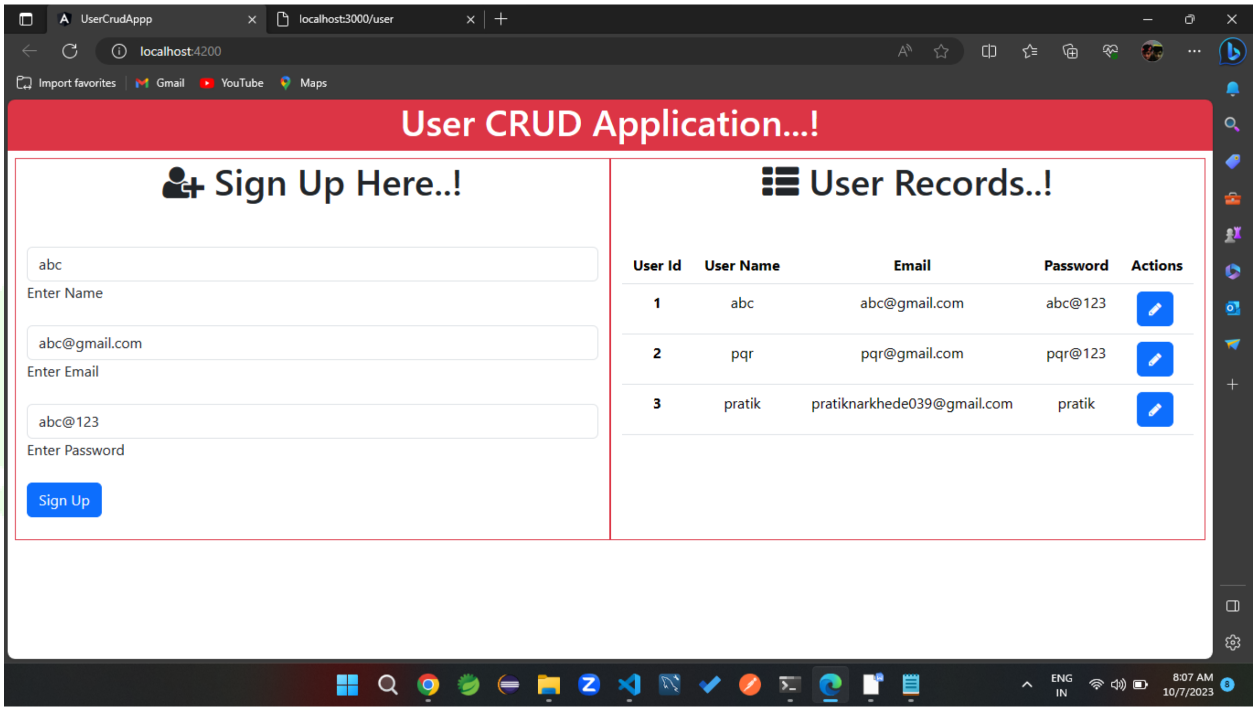


create-user.component.ts U ×

src > app > user > create-user > create-user.component.ts > CreateUserComponent > ngOnChanges

```
import { Component, Input, OnInit, OnChanges, SimpleChanges } from '@angular/core';
import { FormBuilder,FormGroup } from '@angular/forms';
import { User } from 'src/app/model/user';
import { UserService } from 'src/app/service/user.service';
@Component({
  selector: 'app-create-user',
  templateUrl: './create-user.component.html',
  styleUrls: ['./create-user.component.css']
})
export class CreateUserComponent implements OnInit,OnChanges{

  constructor(private fb:FormBuilder,private us:UserService){  }
   userReg:FormGroup;

  @Input() userToBeEdit:User;

  ngOnInit(): void {
     this.userReg=this.fb.group(
      {
        id:[0],
        name:[],
        emailId:[],
        password:[]
      });
  }
  ngOnChanges(){
      if(this.userReg!=null)
      {
      this.userReg.patchValue(
        {
          id:this.userToBeEdit.id,
          name:this.userToBeEdit.name,
          emailId:this.userToBeEdit.emailId,
          password:this.userToBeEdit.password
        }
      );
      }
  }
  onSubmitUserForm()
  {
    this.us.saveUser(this.userReg.value).subscribe();
    this.userReg.reset();
  }
}
```

8. After clicking edit button of any of the row our out put should look like as shown below.



User CRUD Application...!

Sign Up Here..!

abc
Enter Name

abc@gmail.com
Enter Email

abc@123
Enter Password

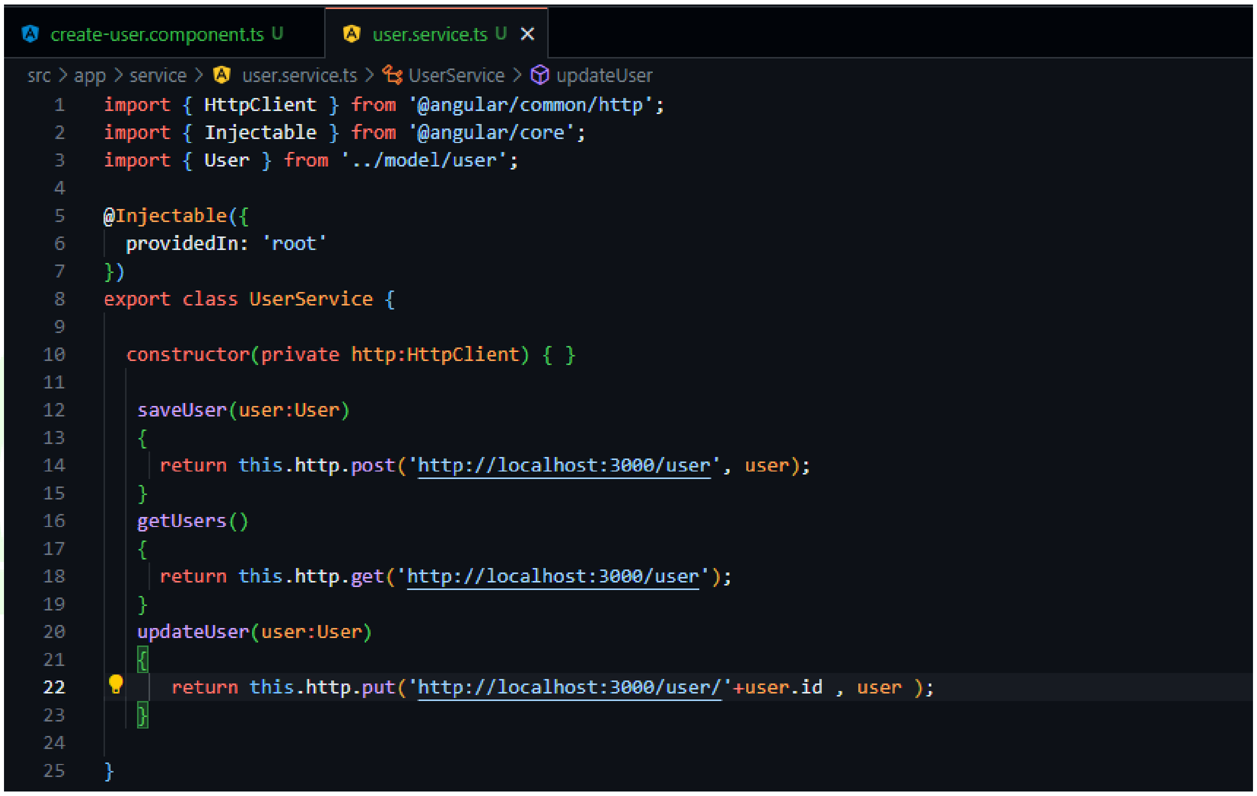
Sign Up

User Records..!

User Id	User Name	Email	Password	Actions
1	abc	abc@gmail.com	abc@123	
2	pqr	pqr@gmail.com	pqr@123	
3	pratik	pratiknarkhede039@gmail.com	pratik	

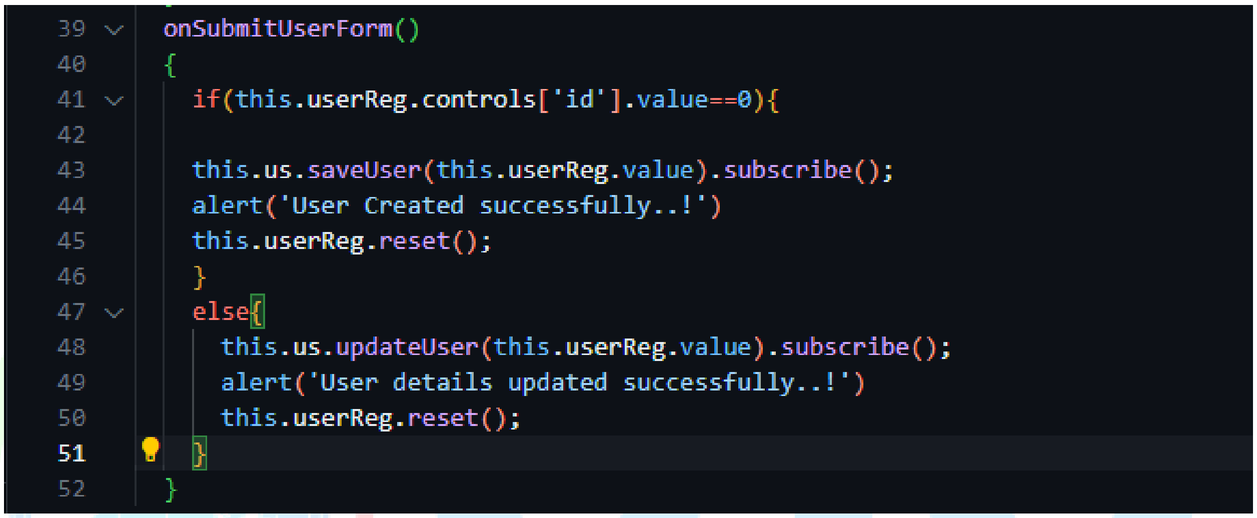
C R U D G E T P U T D E L E T E

9. In UserService declare updateUser function with parameter which will accept User object , And pass it to HttpClient's put function along with end-point url .


create-user.component.ts U
user.service.ts U
src > app > service > user.service.ts > UserService > updateUser

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { User } from '../model/user';

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

  constructor(private http:HttpClient) { }

   saveUser(user:User)
   {
     return this.http.post('http://localhost:3000/user', user);
   }
   getUsers()
   {
     return this.http.get('http://localhost:3000/user');
   }
   updateUser(user:User)
   {
      return this.http.put('http://localhost:3000/user/'+user.id , user );
   }

}
```

10. Now implement one condition in onSubmitUserForm function of CreateUserComponet ,
That if id control of a form have value zero then only save user record otherwise update user record.



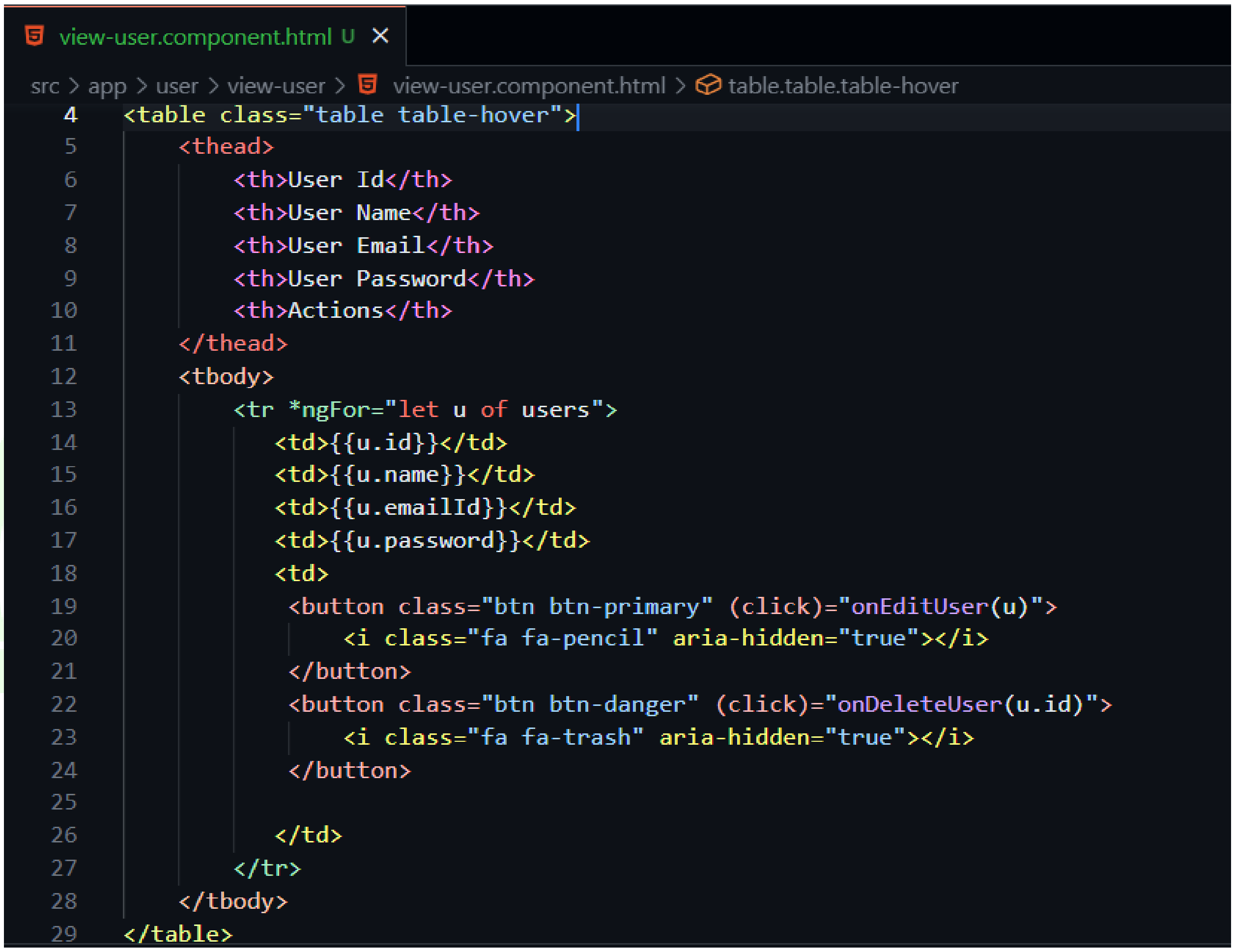
```
onSubmitUserForm()
{
  if(this.userReg.controls['id'].value==0){

  this.us.saveUser(this.userReg.value).subscribe();
  alert('User Created successfully..!')
  this.userReg.reset();
  }
  else{
    this.us.updateUser(this.userReg.value).subscribe();
    alert('User details updated successfully..!')
    this.userReg.reset();
  }
}
```

Note:- Now try to edit user details on browser.

C R U D G E T P U T D E L E T E

❖ Steps to delete existing user record :-

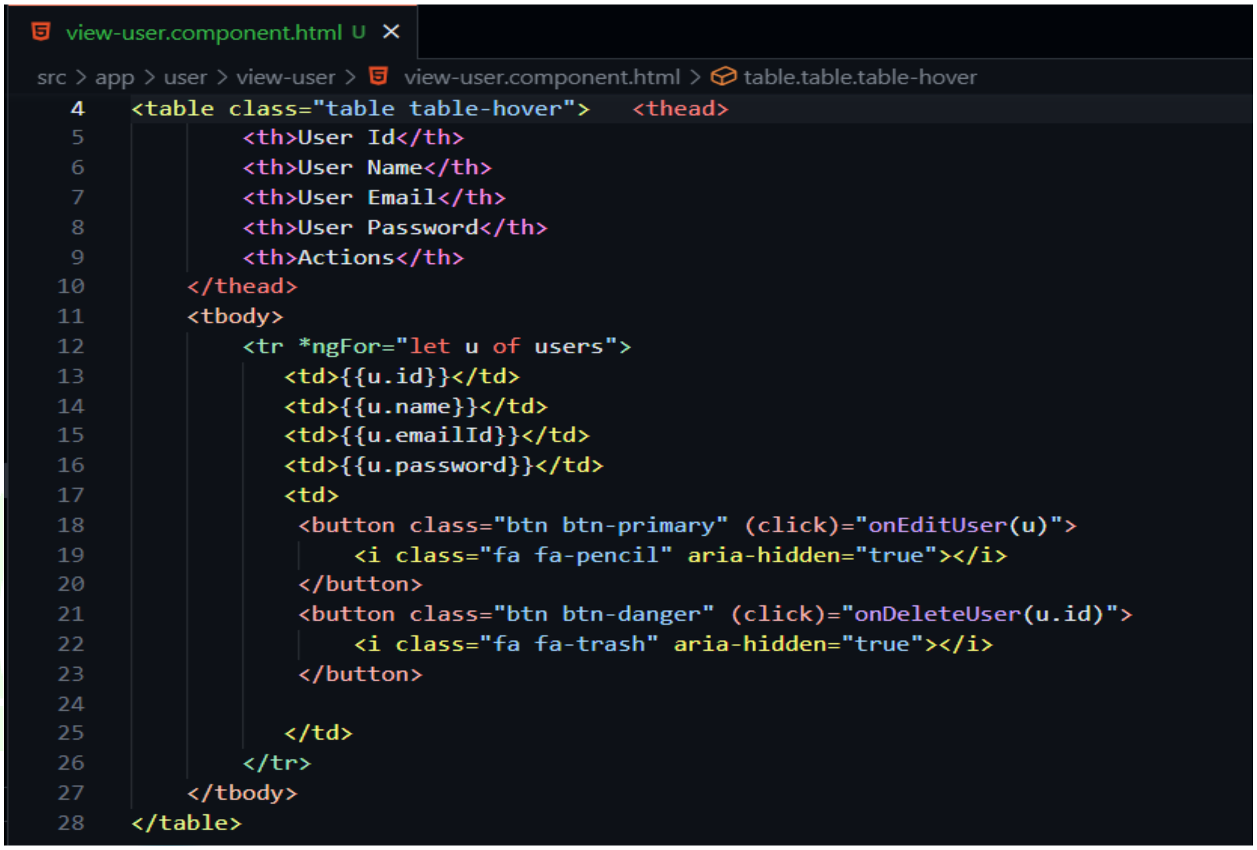
1. In the view-user.component.html add one more button to delete the user record also bind one function on click event of button.


view-user.component.html U ×
src > app > user > view-user > view-user.component.html > table.table.table-hover

```html
<table class="table table-hover">
    <thead>
        <th>User Id</th>
        <th>User Name</th>
        <th>User Email</th>
        <th>User Password</th>
        <th>Actions</th>
    </thead>
    <tbody>
        <tr *ngFor="let u of users">
           <td>{{u.id}}</td>
           <td>{{u.name}}</td>
           <td>{{u.emailId}}</td>
           <td>{{u.password}}</td>
           <td>
            <button class="btn btn-primary" (click)="onEditUser(u)">
                <i class="fa fa-pencil" aria-hidden="true"></i>
            </button>
            <button class="btn btn-danger" (click)="onDeleteUser(u.id)">
                <i class="fa fa-trash" aria-hidden="true"></i>
            </button>

           </td>
        </tr>
    </tbody>
</table>
```

C R U D G E T P U T D E L E T E

2. In ViewUserComponent declare same function that you have bind to the delete button add in that function call deleteUser unction of a UserService.



view-user.component.html U ×

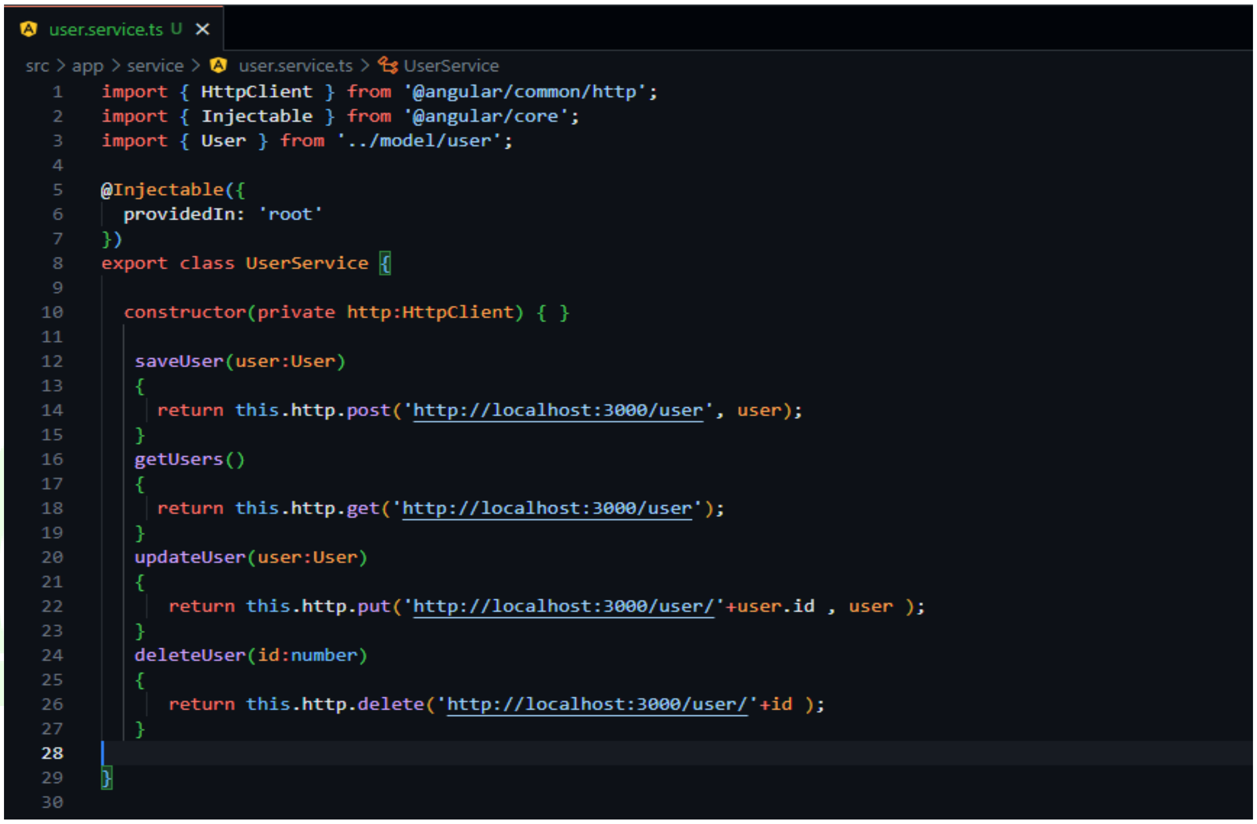
src > app > user > view-user > view-user.component.html > table.table.table-hover

```html
<table class="table table-hover">   <thead>
        <th>User Id</th>
        <th>User Name</th>
        <th>User Email</th>
        <th>User Password</th>
        <th>Actions</th>
    </thead>
    <tbody>
        <tr *ngFor="let u of users">
            <td>{{u.id}}</td>
            <td>{{u.name}}</td>
            <td>{{u.emailId}}</td>
            <td>{{u.password}}</td>
            <td>
             <button class="btn btn-primary" (click)="onEditUser(u)">
                 <i class="fa fa-pencil" aria-hidden="true"></i>
             </button>
             <button class="btn btn-danger" (click)="onDeleteUser(u.id)">
                 <i class="fa fa-trash" aria-hidden="true"></i>
             </button>

            </td>
        </tr>
    </tbody>
</table>
```

by Kunal Sir

C R U D G E T P U T D E L E T E

3. Create one deleteUser function in UserService which will accept id type of number as a parameter argument and consume delete end-point of a json-server.



```typescript
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { User } from '../model/user';

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

  constructor(private http:HttpClient) { }

   saveUser(user:User)
   {
     return this.http.post('http://localhost:3000/user', user);
   }
   getUsers()
   {
     return this.http.get('http://localhost:3000/user');
   }
   updateUser(user:User)
   {
      return this.http.put('http://localhost:3000/user/'+user.id , user );
   }
   deleteUser(id:number)
   {
      return this.http.delete('http://localhost:3000/user/'+id );
   }

}
```