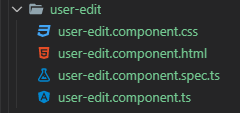


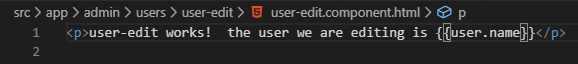
Create component for editing users:

>ng g c admin/users/UserEdit

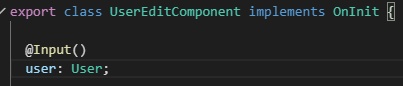




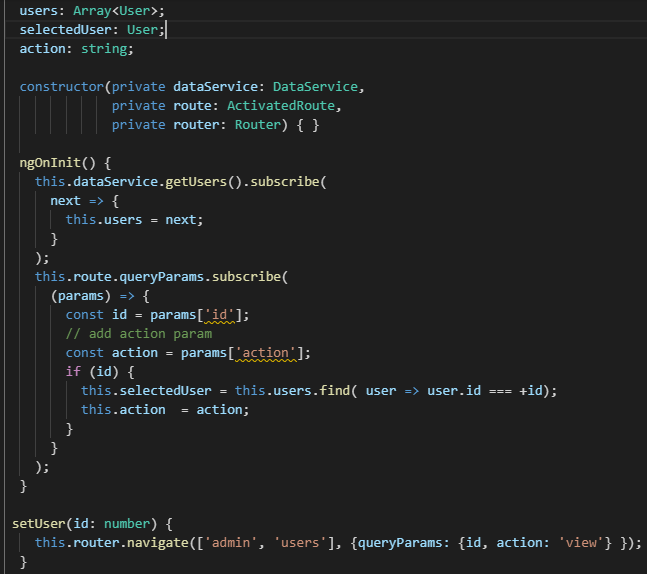
Set up a call to output the user in the user-edit.component.ts file



Using navigation as well



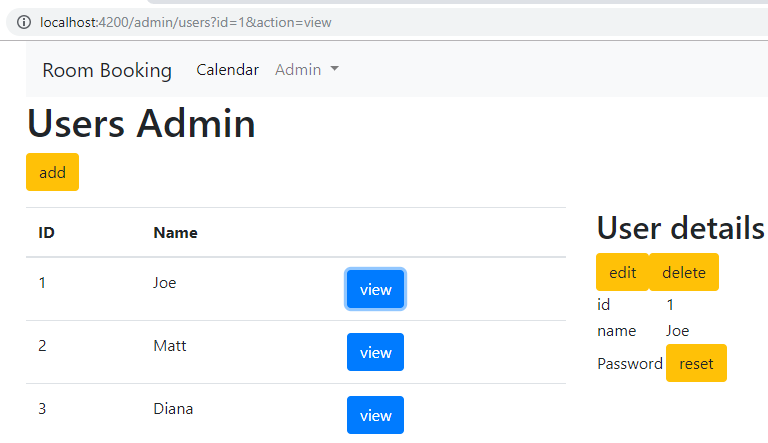
In the user.component.ts file we create action and use it in setUser method



In the user.component.html, we only want user-detail to show if view is selected



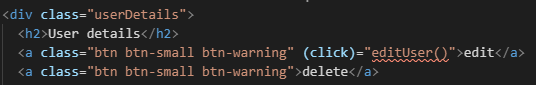
Now, test that it still works

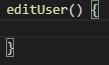


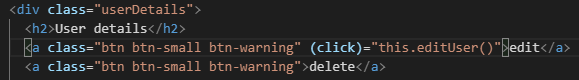
Add the edit version to the html



Add a click event to the Edit button



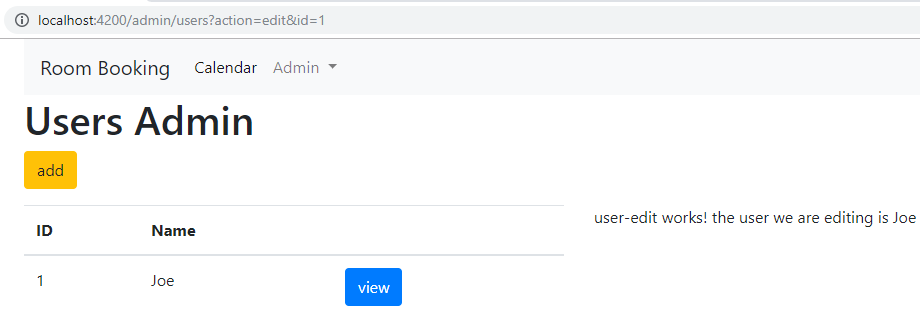




Navigate to the correct url to edit the user – this is what goes into the editUser() method



Test clicking edit button - Joe and the navigation admin/users?action=edit&id=1 works!!



**Now, let us complete the Form**

Let us get the html from design and added to the user-edit.component.html file

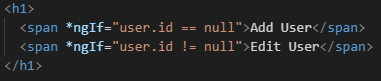


Fill in the place holders

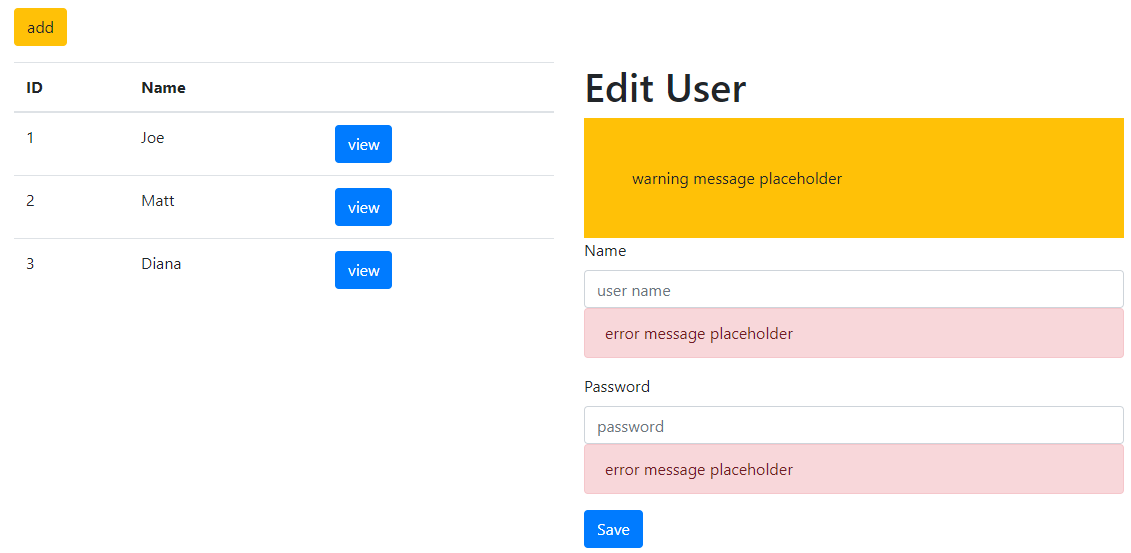
Null user is ADD mode

Not null user is EDIT mode

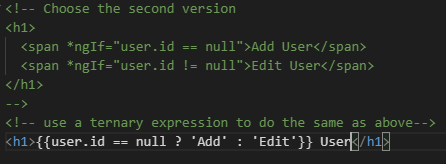
Create 2 spans each one of the cases above



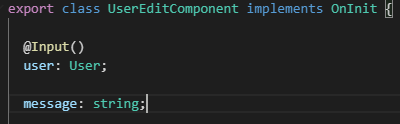
Test if the Edit screen shows up



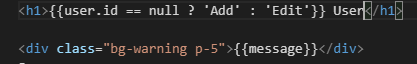
There is a ternary version that takes less code:



Now, let us add a message in class user-edit.component.ts file



In the html component or user-edit.component.html file



Now, the message appears blank



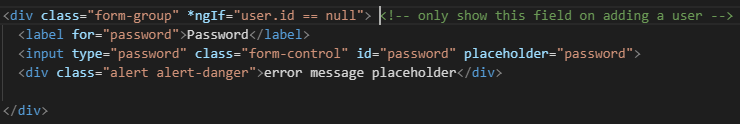
When is message is blank, it should not appear. So, if message !== null it will appear.

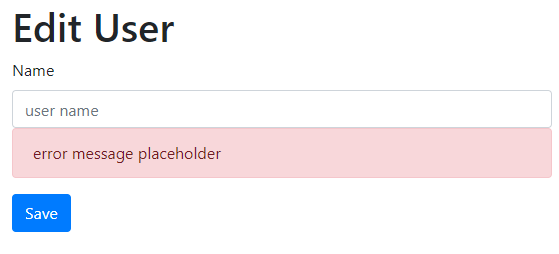
But, this is the same as if(message) in javascript. If(null) is false





Now, let us use \*ngIf to only show the password boxes when Add is selected





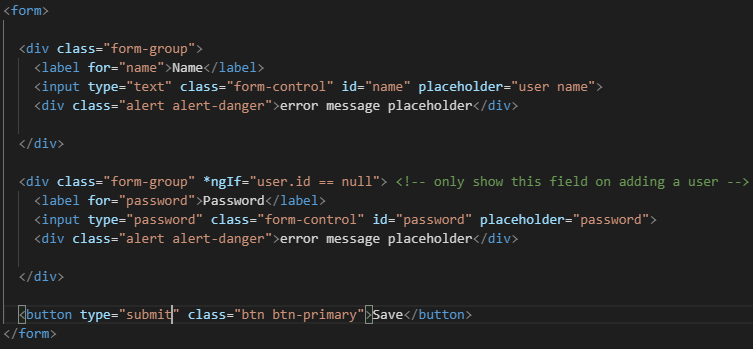
**Let us bind our form**

Remove the get and post info in the form

Create onSubmit() but do not use type=submit or the page will refresh ( on the user-edit.component.ts )



Use type=button instead



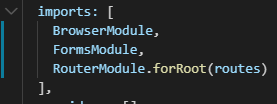
Remove type=”submit”



Need inputs binded with variables

Let us set up the app.module.ts file for this

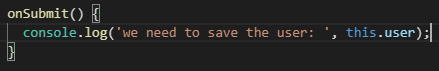




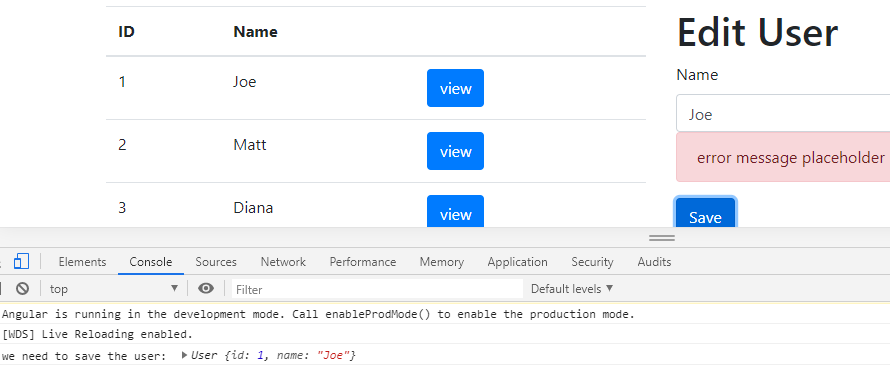
Now, we are going to use 2-way bind [(user)], we can read from backend and write to the backend



Use a console.log to check the passing of the values in the onSubmit()



Test it



As you type in the Name field the name in the table changes immediately.

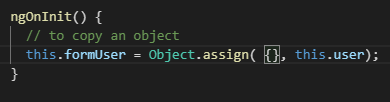
To avoid this create an intermediate user. formUser: User;

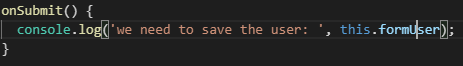


Use the formUser in the Edit name field

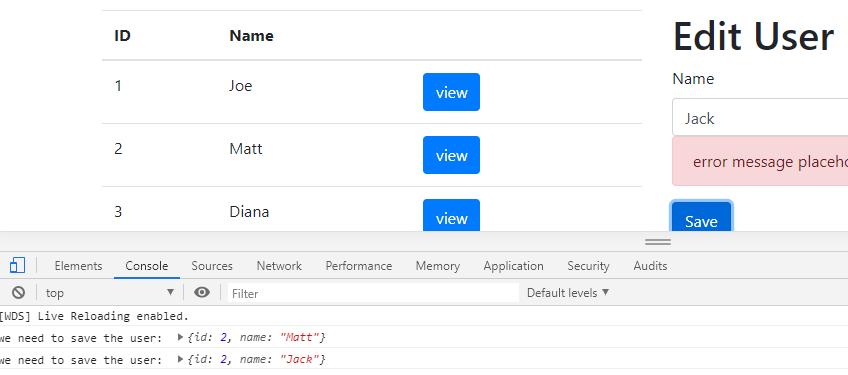


To create an Object to copy to and from





Test this

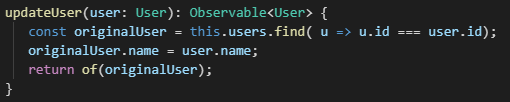


Matt was the initial value in the table and name field on save is Matt

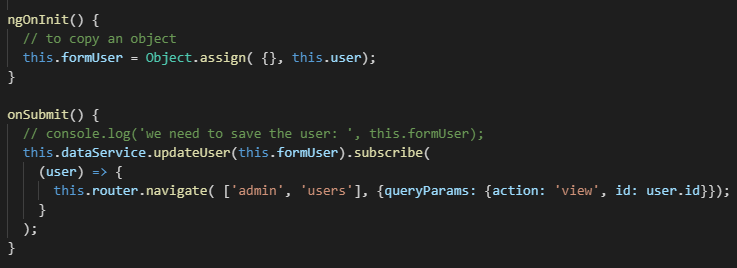
If I change Matt to Jack the table remains unchanged but Jack is not in the onSubmit and the name field

**Let us save an updated user**

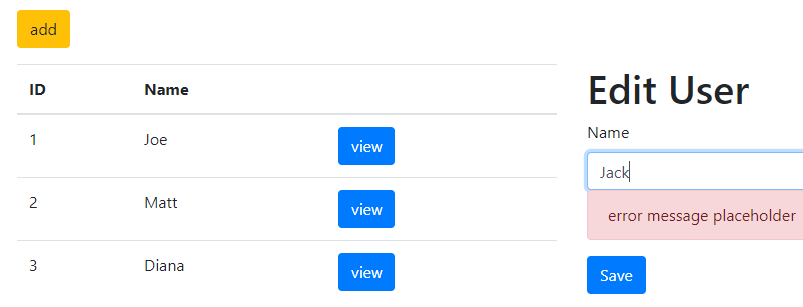
Create updateUser() in the dataService



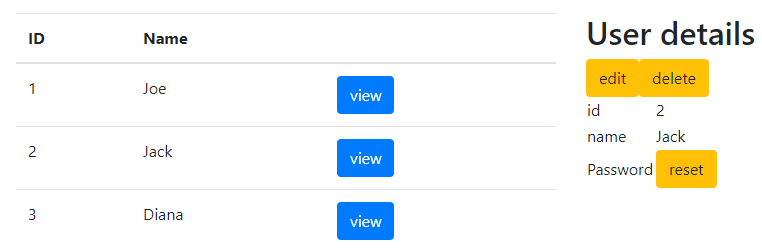
In the user-edit.component.ts file



Test it

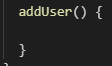


Click save the Matt becomes Jack



**Now, let us fix the Add button**

Add a click event to the users.component.ts



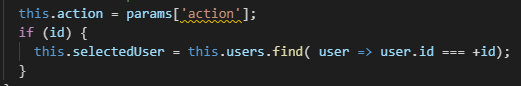
Go to the users.component.html and implement the click event



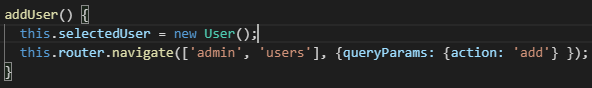
Implement the method navigation



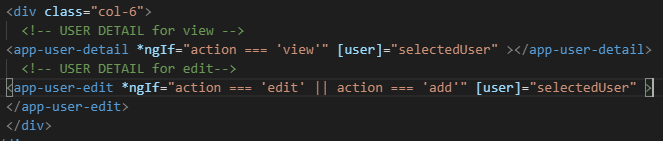
Shorten up the this.action



Create the user in the Add method

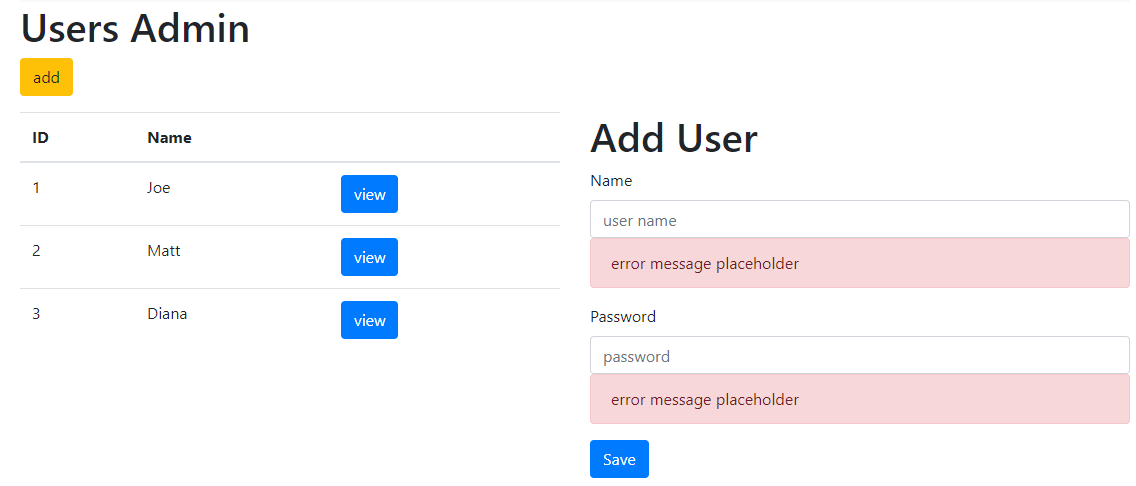


Add the action to the html

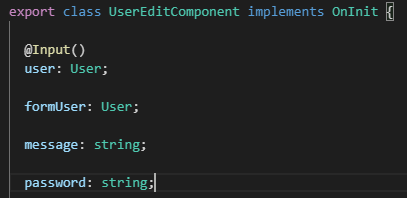


Test it

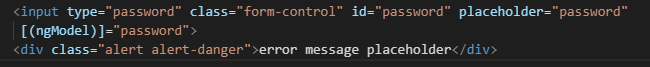
Click on add



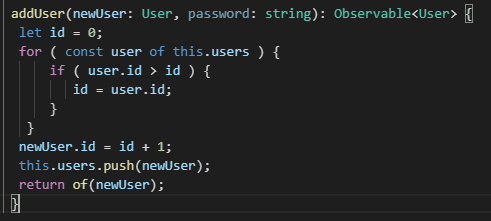
Use a password field and bind that to password



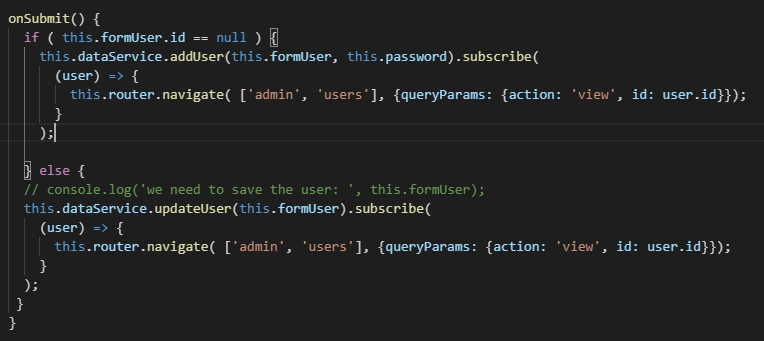
Create the 2-way binding for password



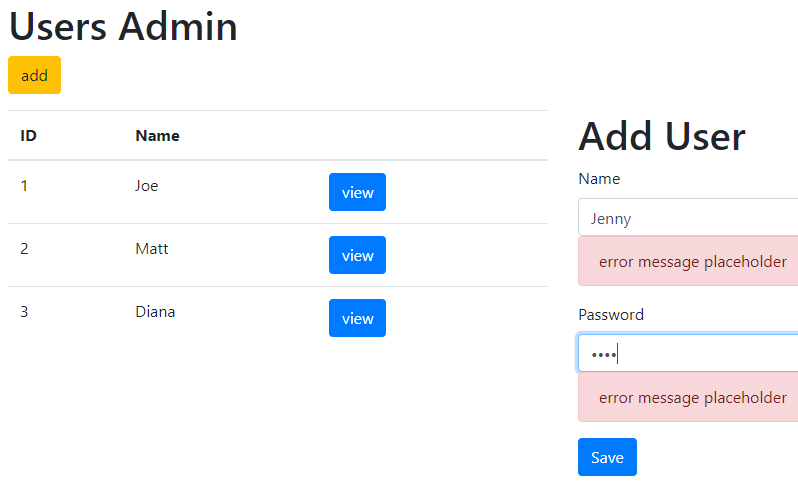
Create the method on save



Now, call it in the onSubmit



Test it



Click Save

