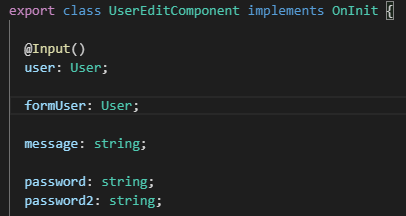
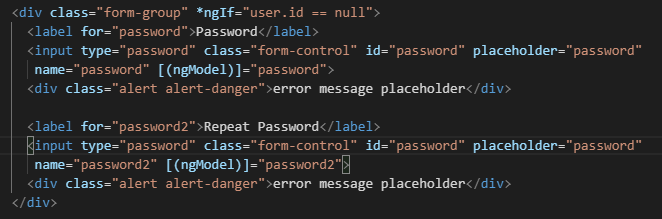
**Validation:**

Blank out the message fields

Create password2 in the user-edit.component.ts file:

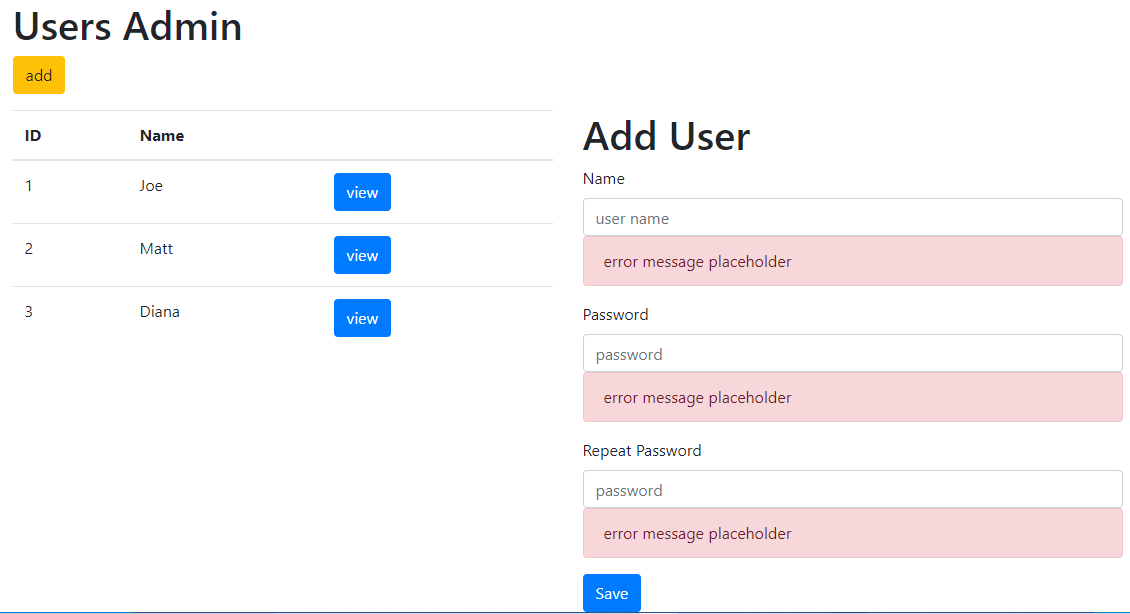


Add password2 to the user-edit.component.html file:

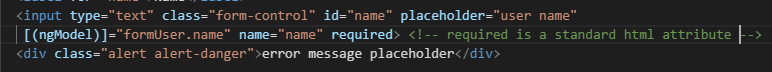


Test it

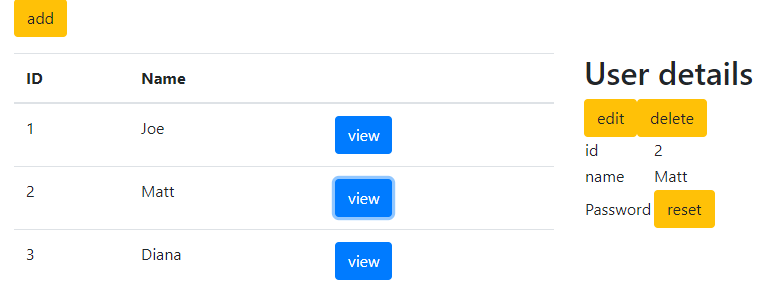
2 password fields when we click on the Add button



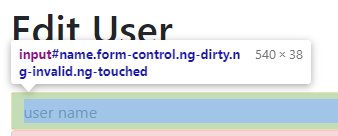
Add required:



Now 4 angular classes are modified when we remove a name:



Dynamic Angular class changes!!

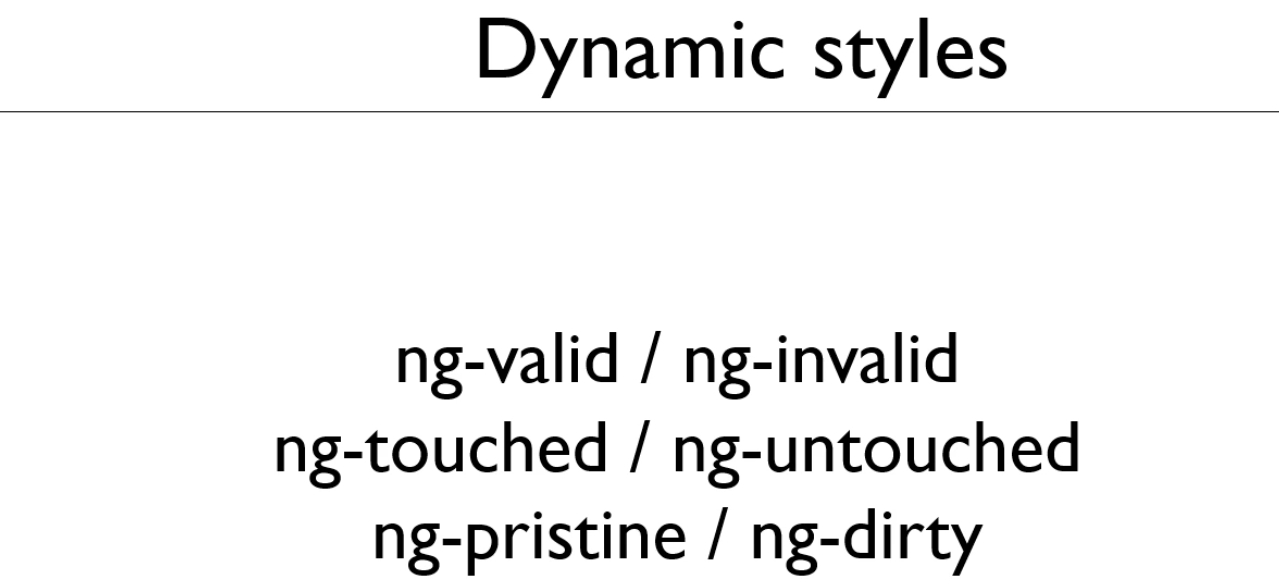


.ng-dirty .ng-invalid .ng-touched classes

Adding the name back Matt



.ng-dirty .ng-touched .ng-valid classes



**.ng-valid – entry of form element passes validation test**

**.ng-invalid – entry of form element failed validation test**

.ng-touched – user has selected the element

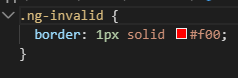
.ng-untouched – user has not selected the element

.ng-pristine – user has not edited the value in the control

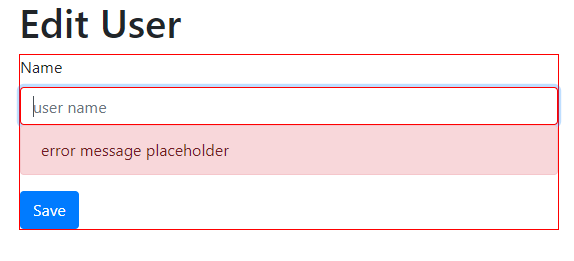
.ng-dirty – user has edited the value in the control

Let us use the CSS file to effect these changes

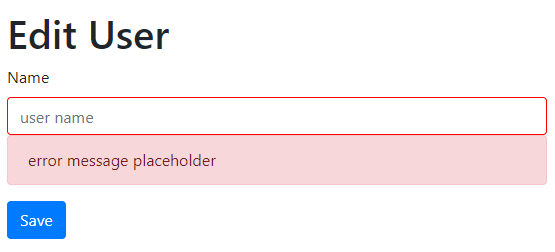
In the user-edit.component.css file



Remove the name and field is invalid



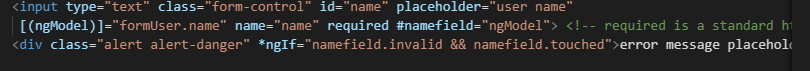
Now tighten up the control by changing the css (touch and untouch and red appears )



Validation messages:

Only so the error message divs if the field is invalid

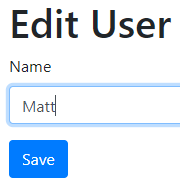
Create a template with a hash #namefield=”ngModel” then use the hash to call classes

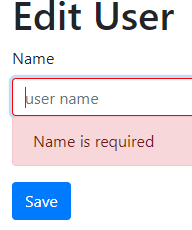


Add a meaningful message



Test it





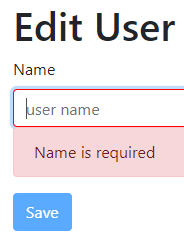
Disable the Save button until the name is filled in

Put the template reference on the form





Test it: Save is lighted showing disabled



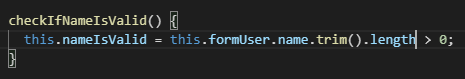
**Custom Validation Rules:**

Spaces is not a name, so let us not allow this

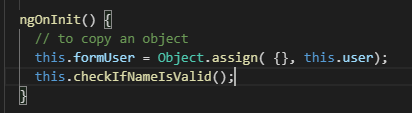
Create variable nameIsValid in the user-edit.component.ts file



Create a method to determine if name is valid in the user-edit.component.ts file

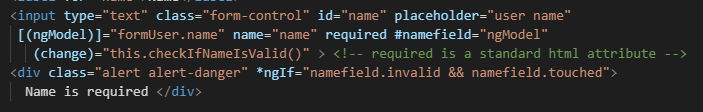


Call the method in the ngOnIt()in the user-edit.component.ts file

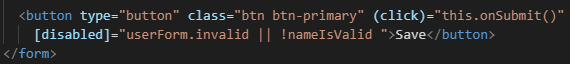


Whenever the value of the name field changes we want to call this method

Bind the change event



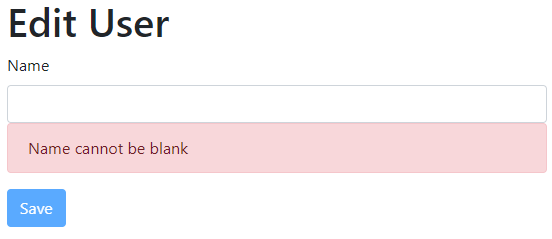
Let us disable the Save button when the nameIsInvalid is false



Add error message “Name cannot be blank”

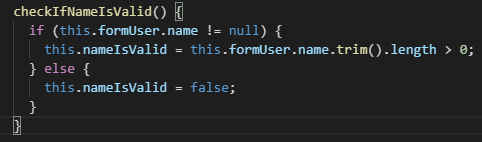


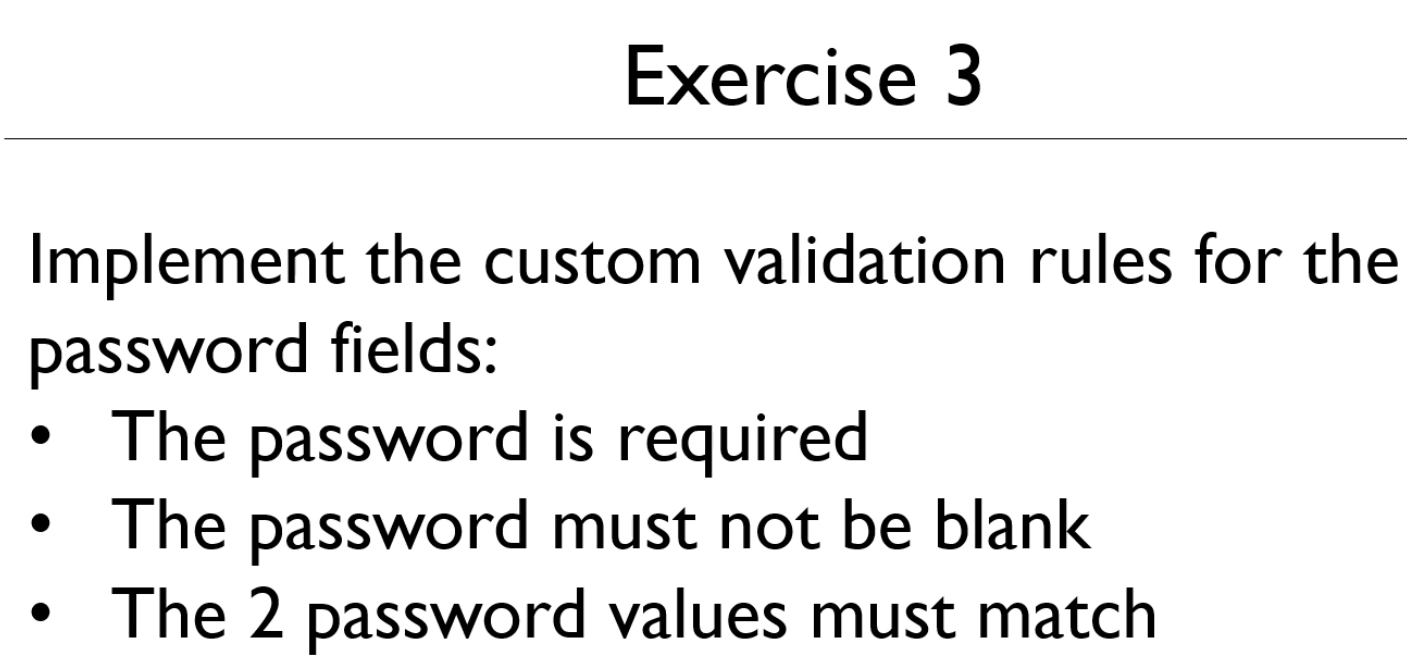
Test this by saving spaces



Notice that if we click on Add we get a trim error message

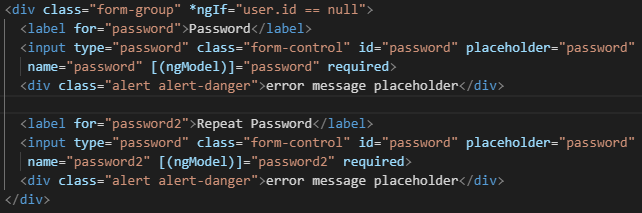
To fix this add a null check, maintain the false



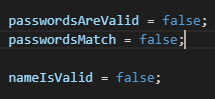


The password is required

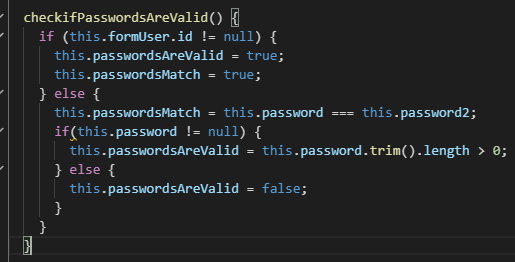
Add required the password input fields



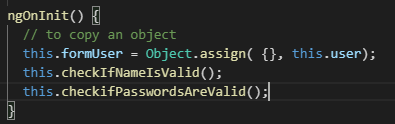
Create variables passwordsAreValid and passwordsMatch



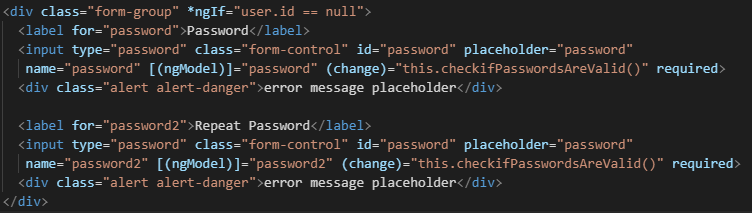
Create a checking method



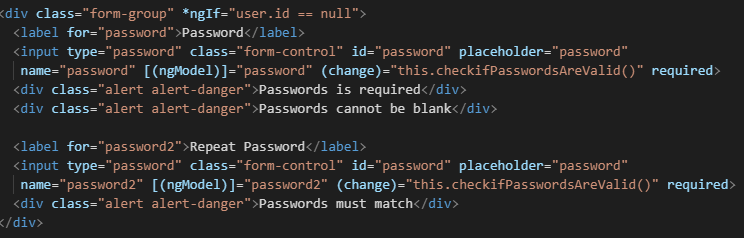
Call the method



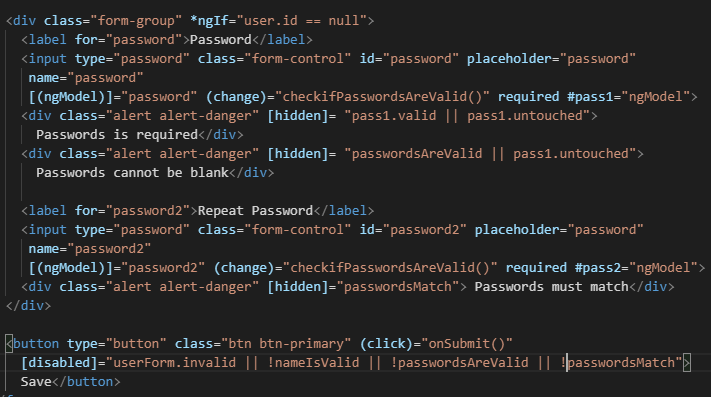
If either password field is edited



Add sensible messages



Disable save button and template reference



Test it

