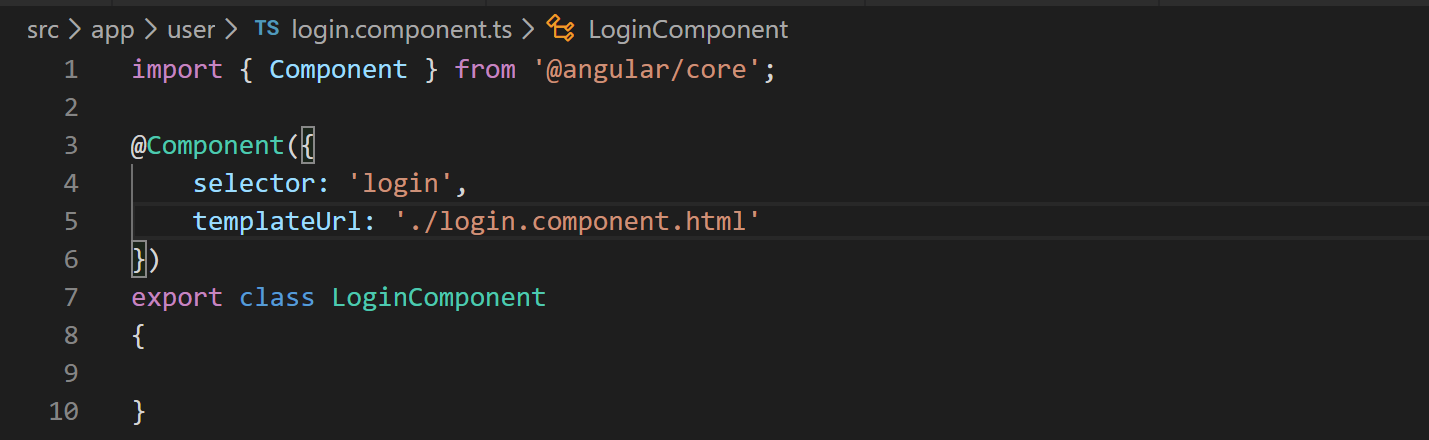
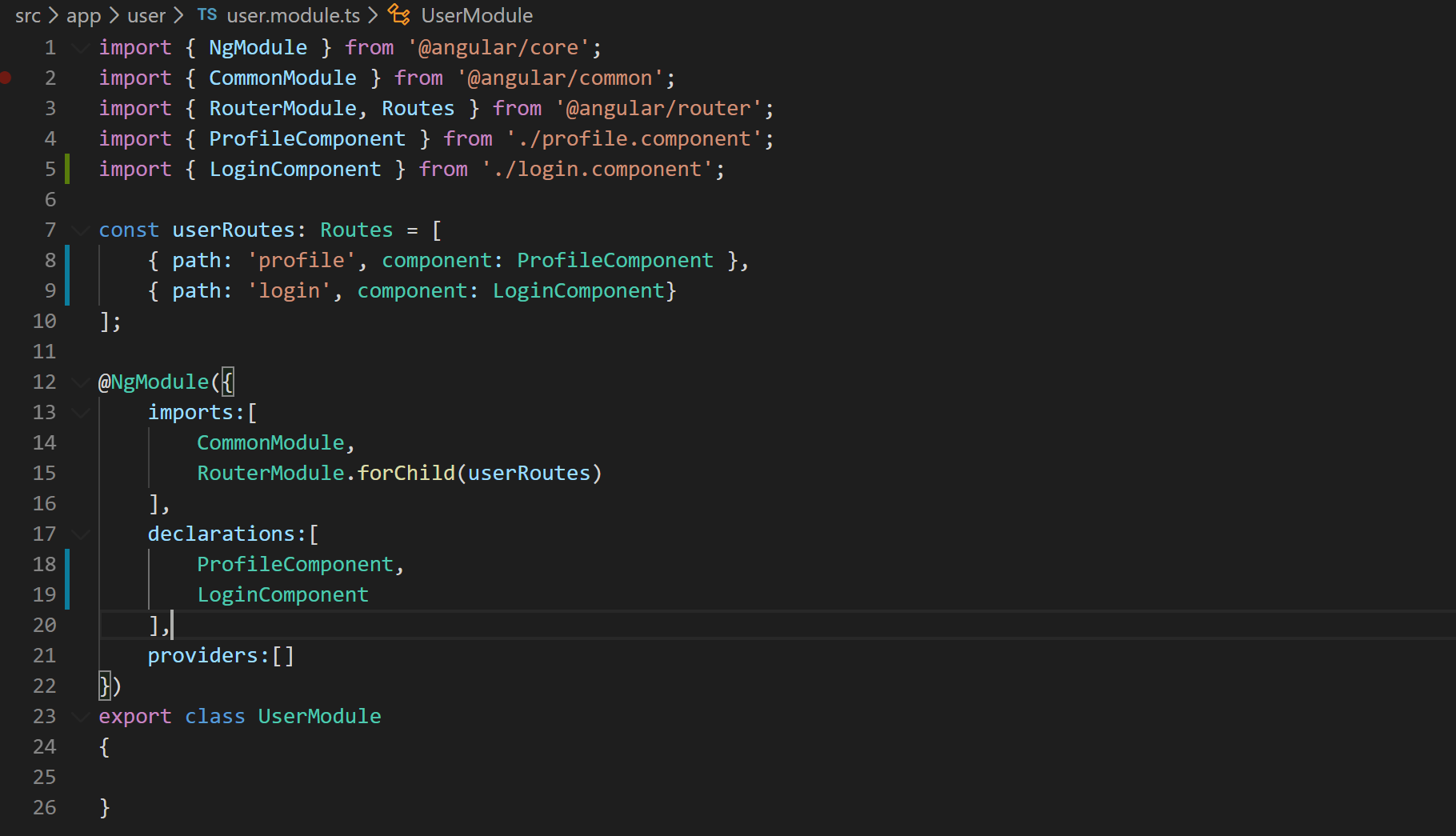
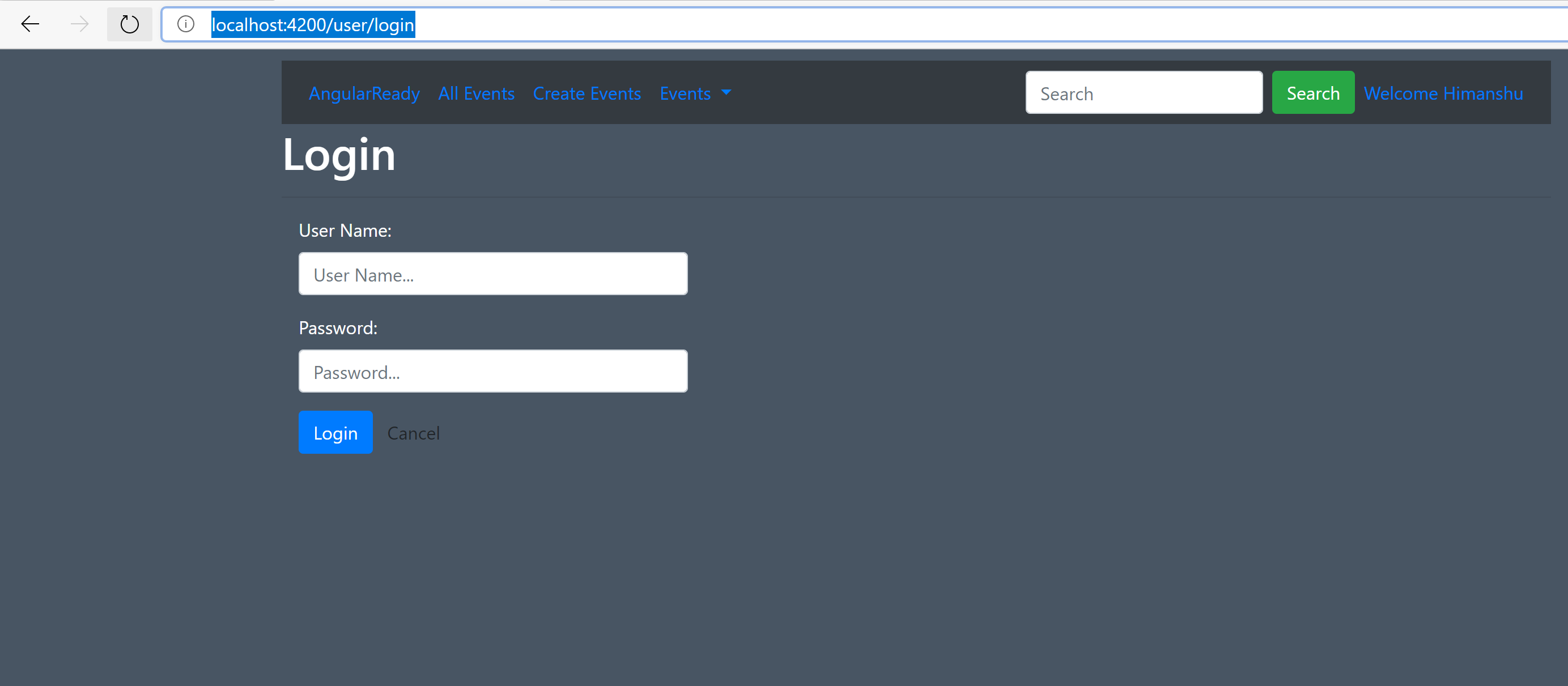
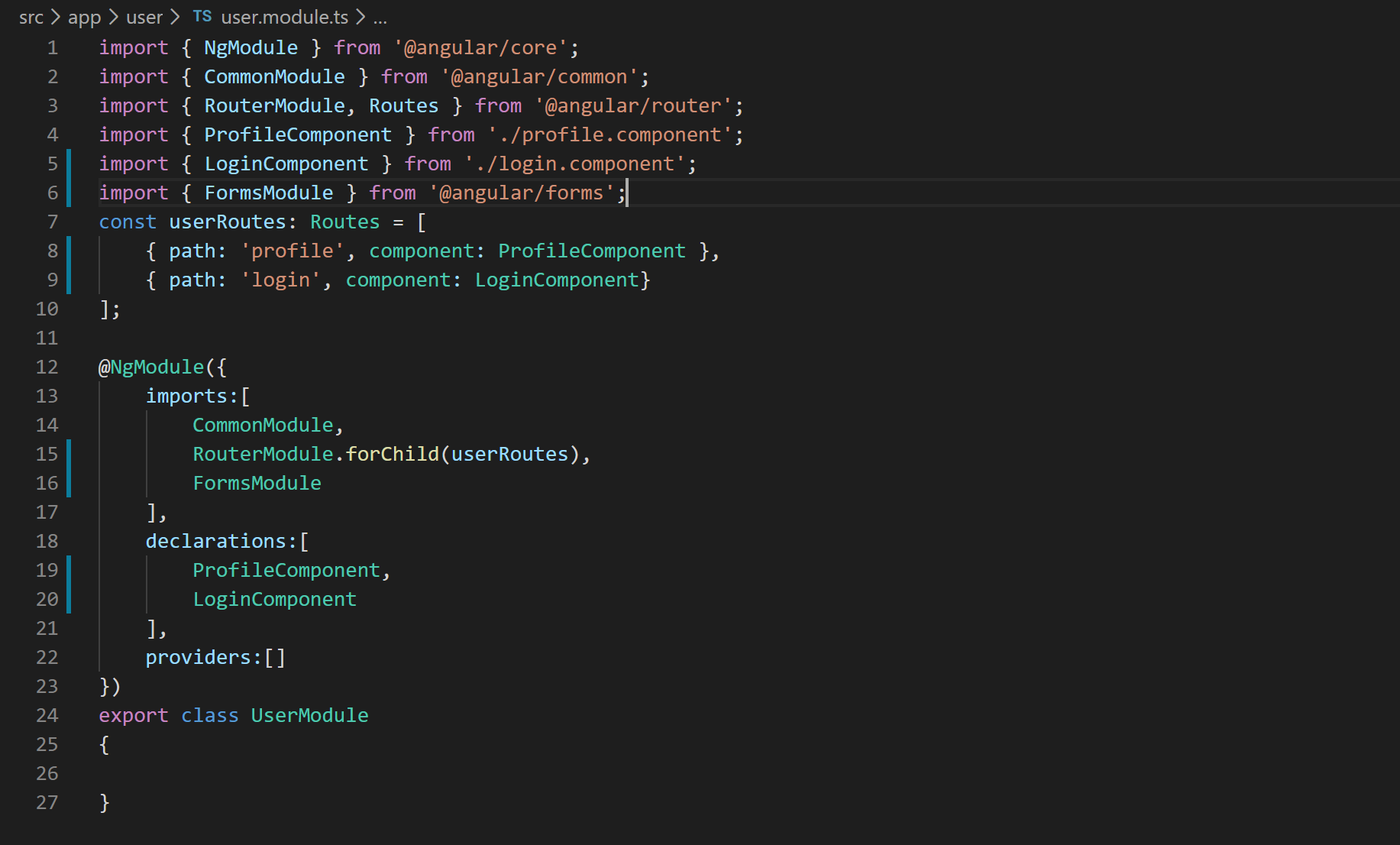
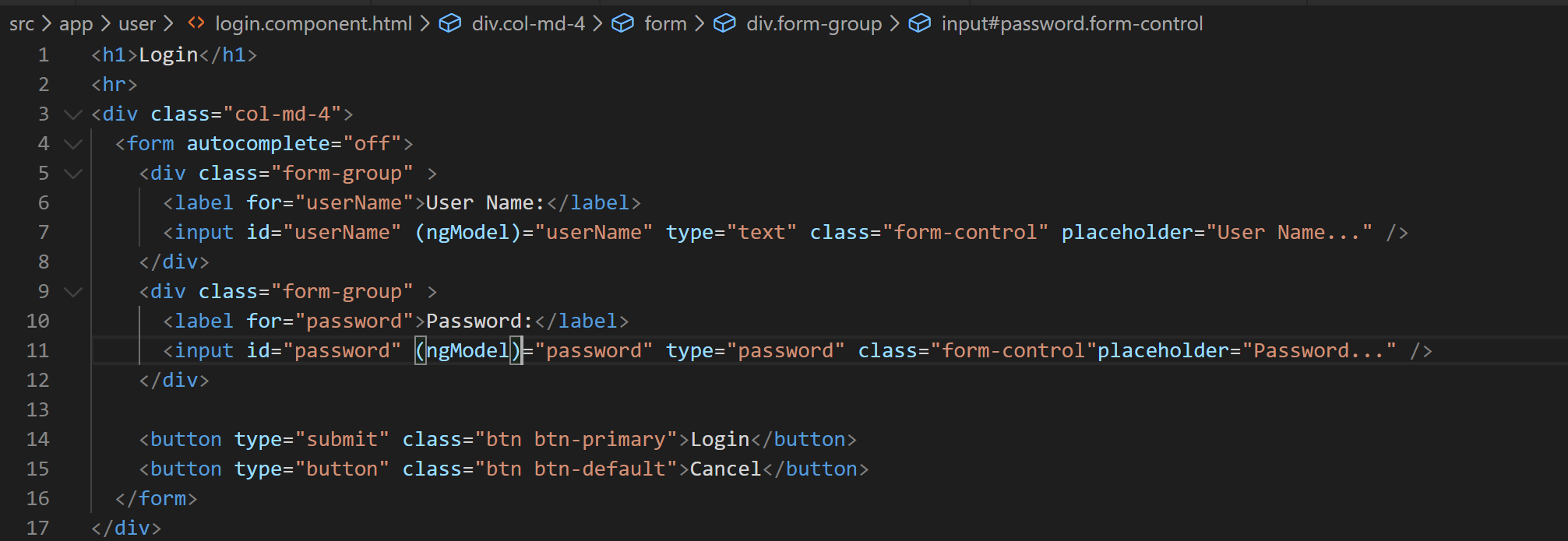
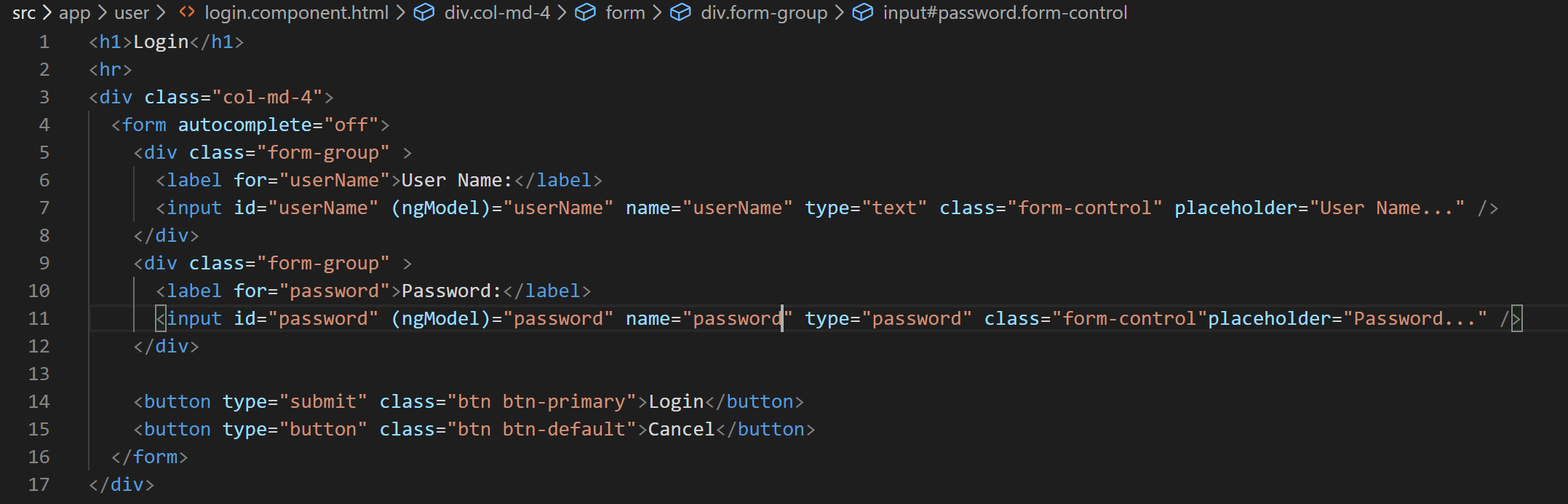
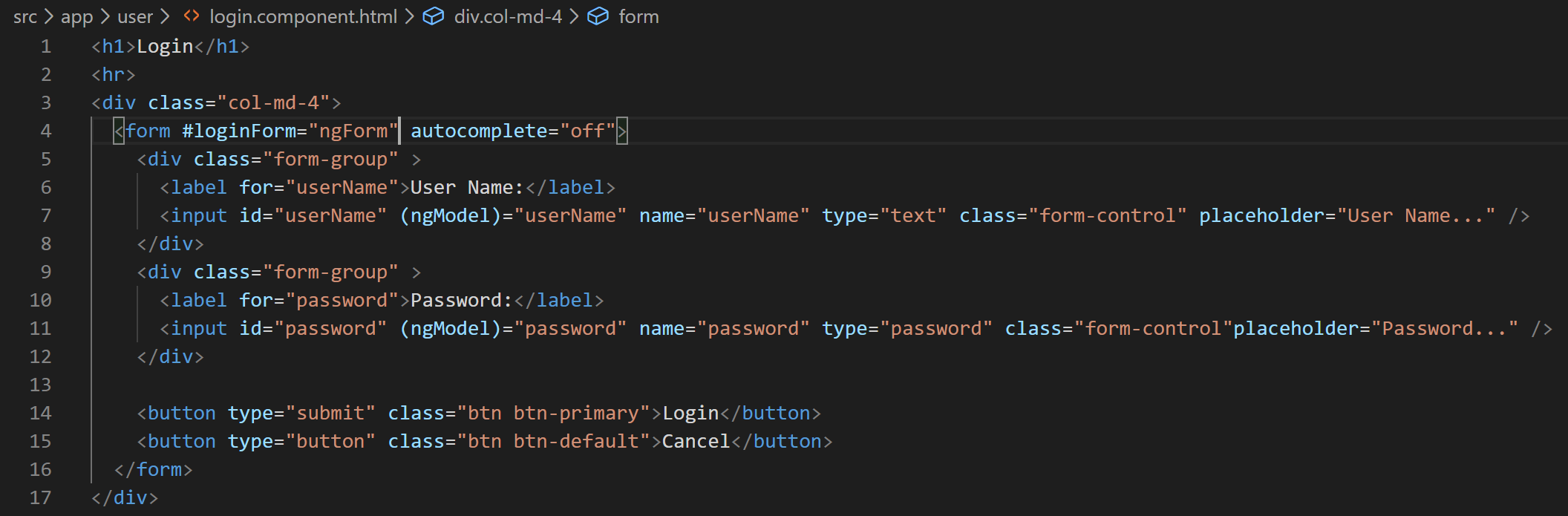
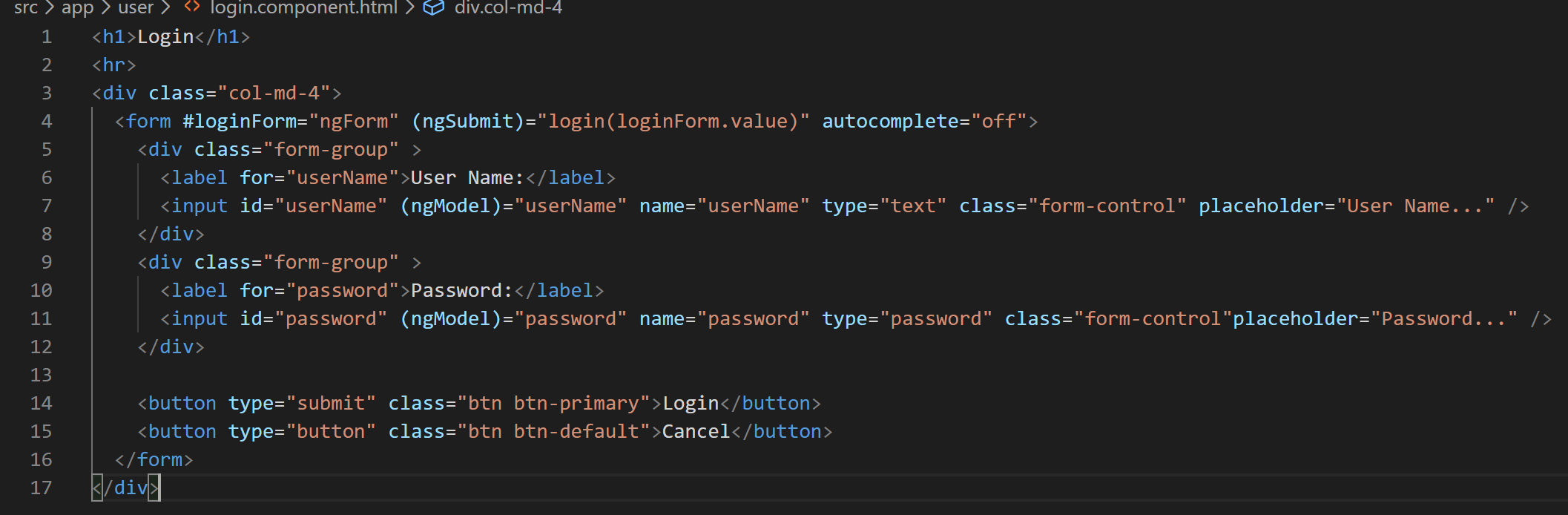
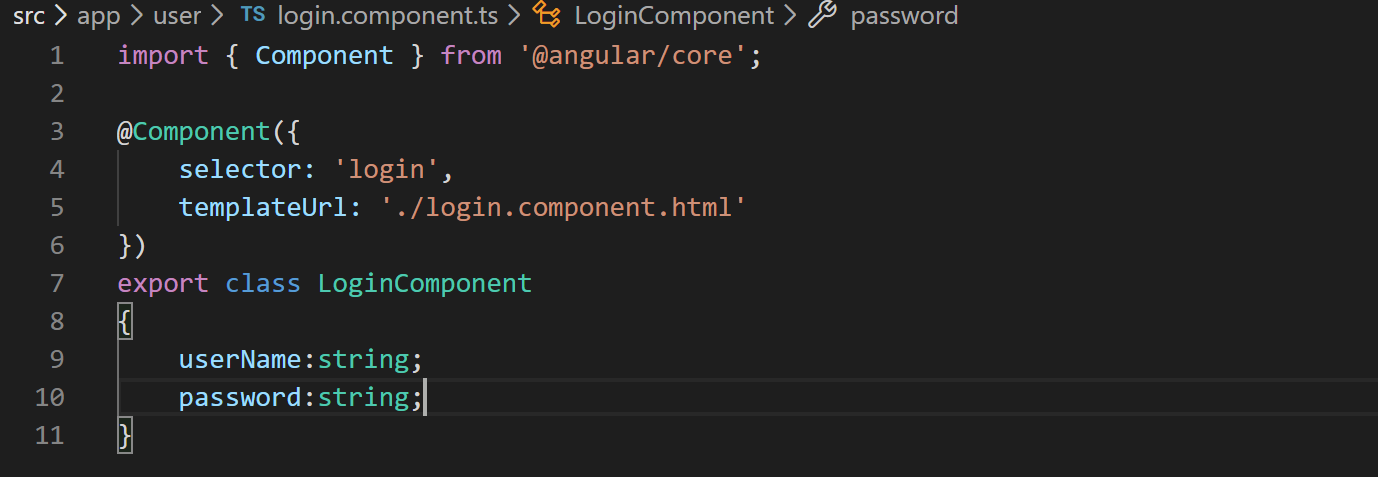
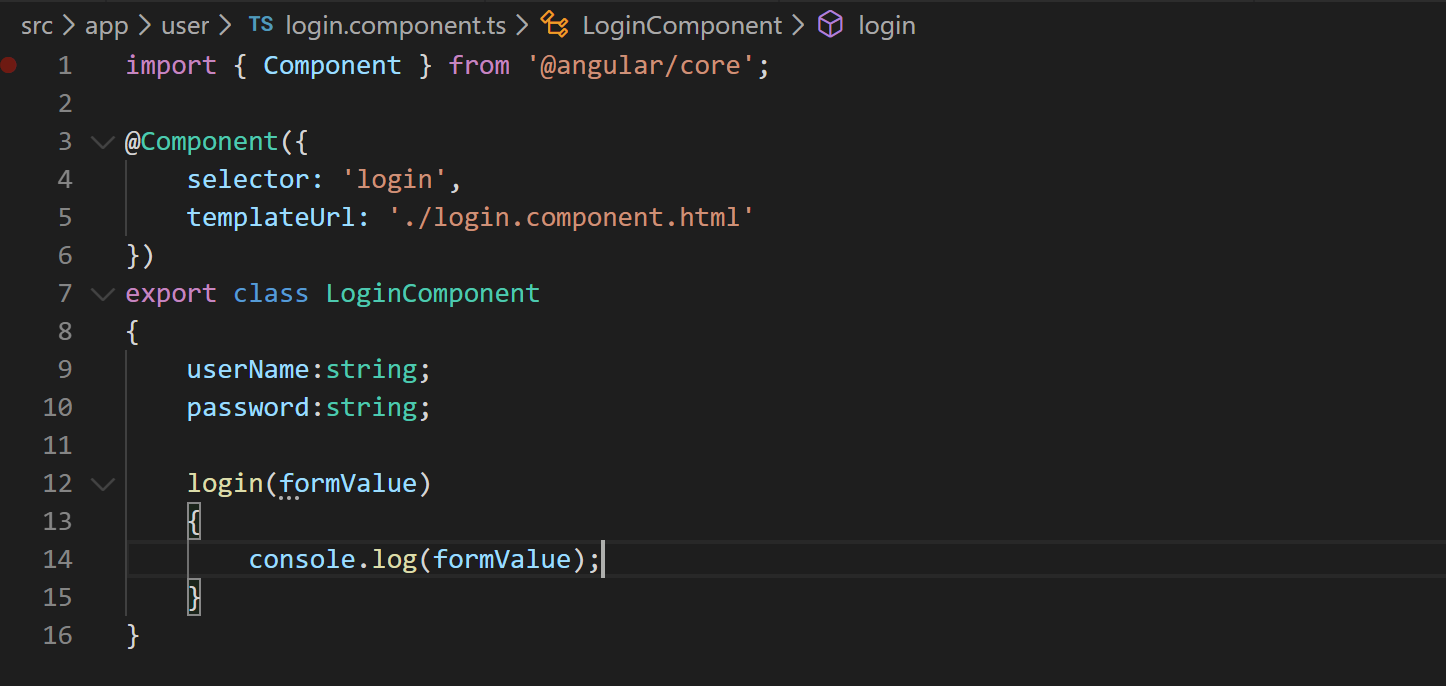
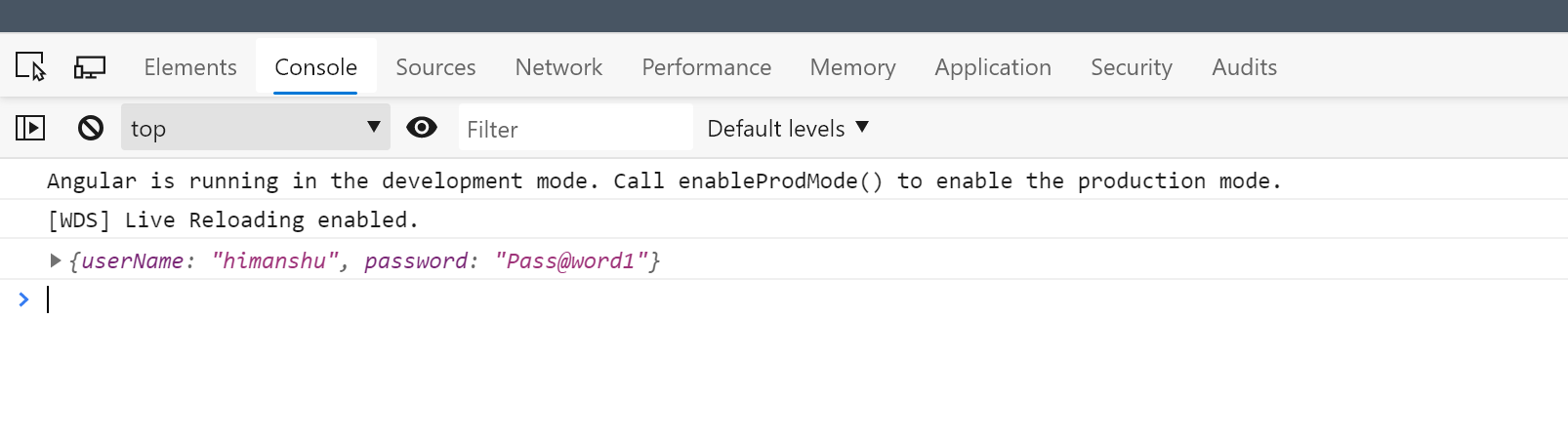
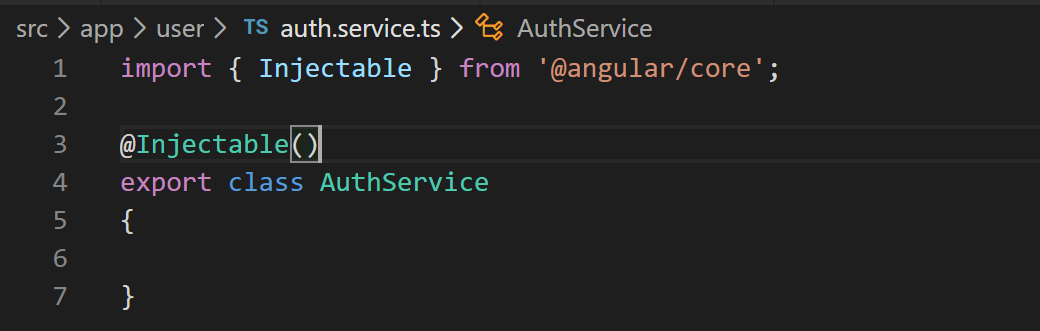
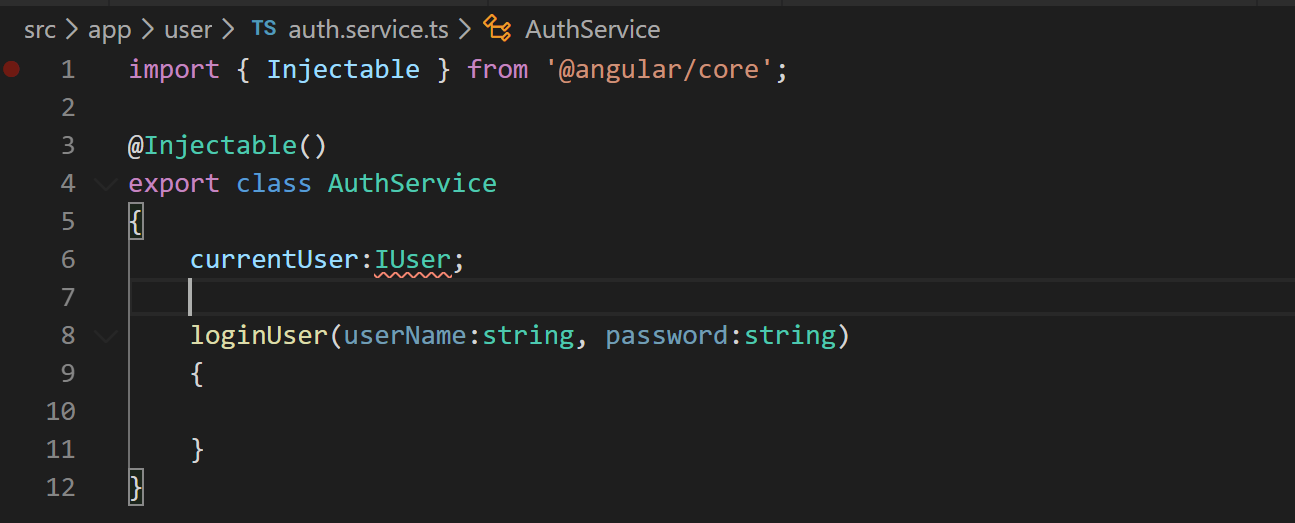
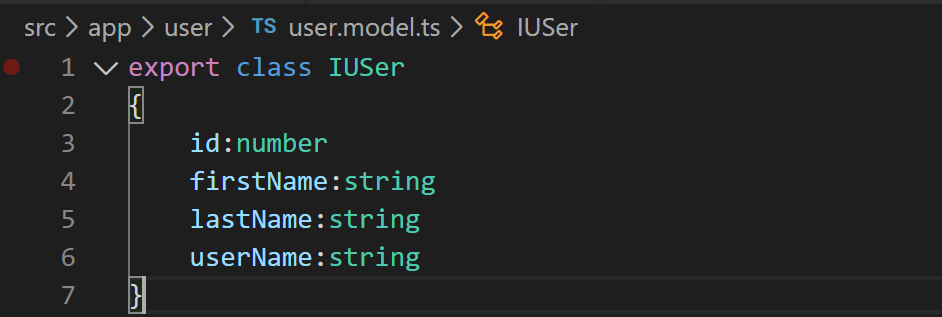
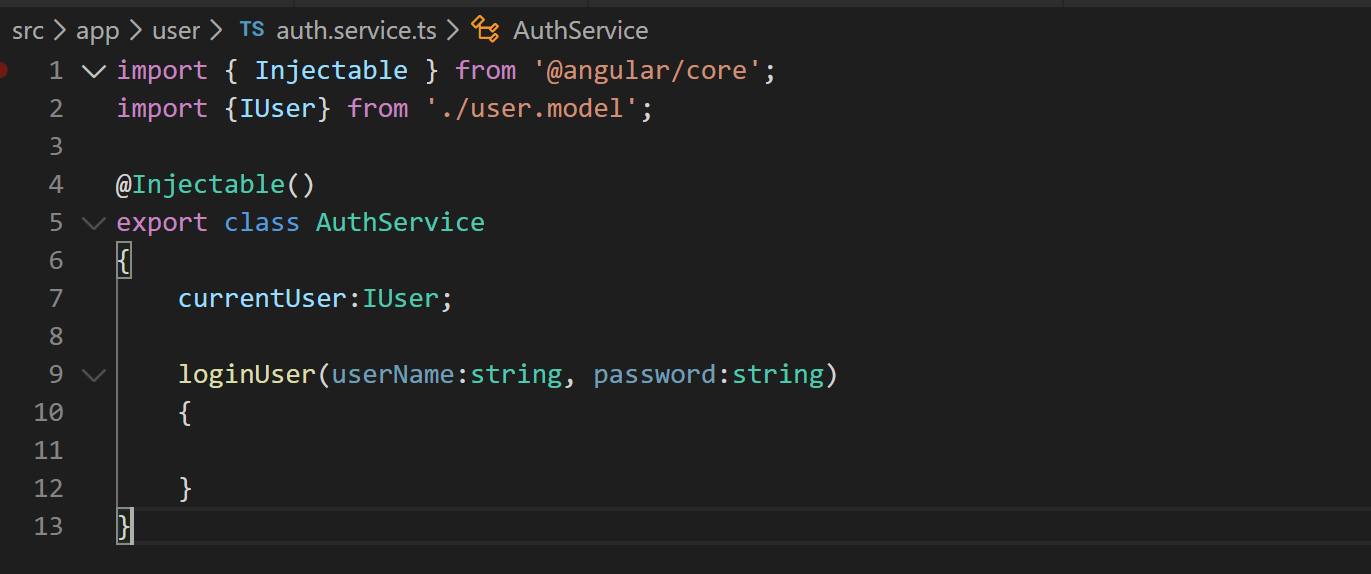
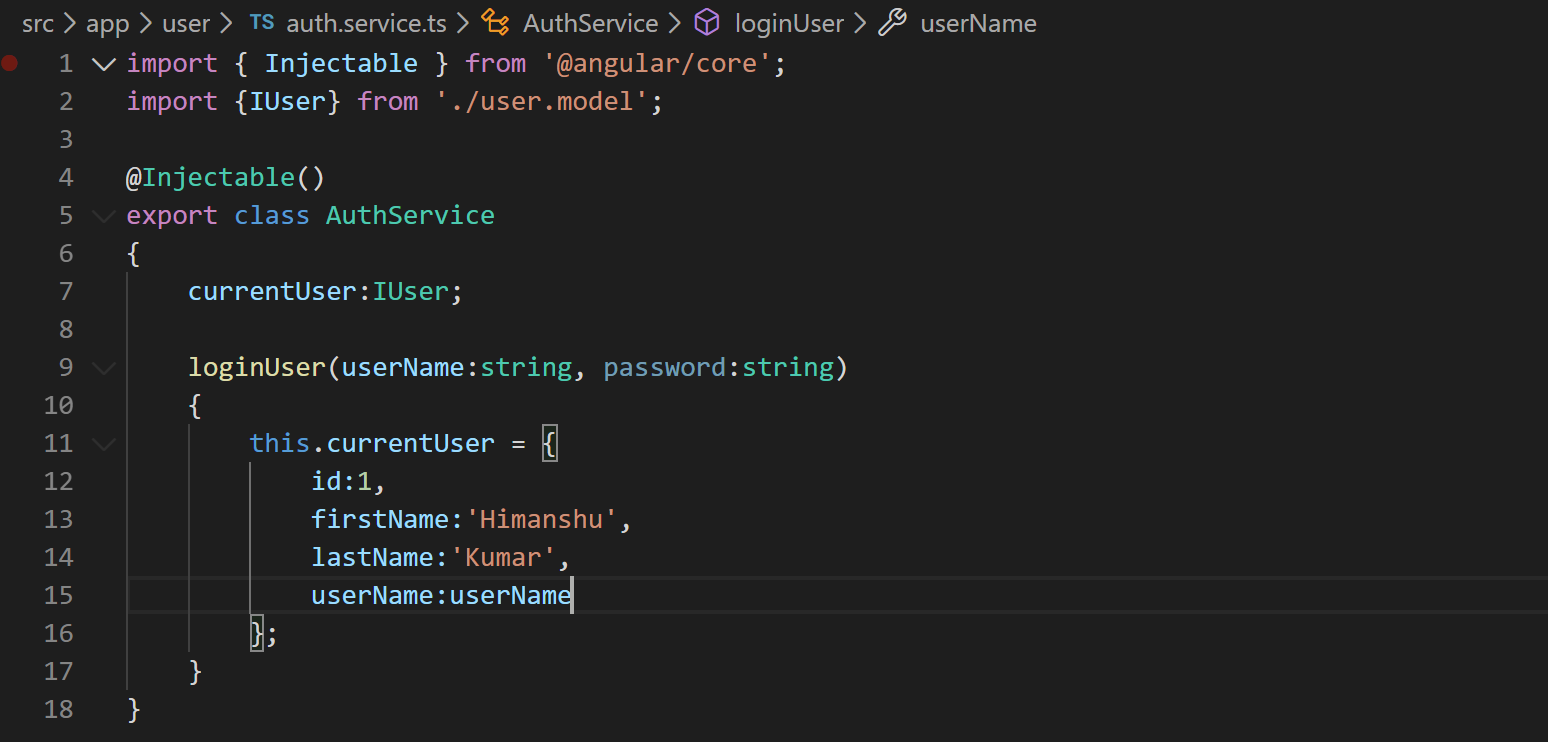
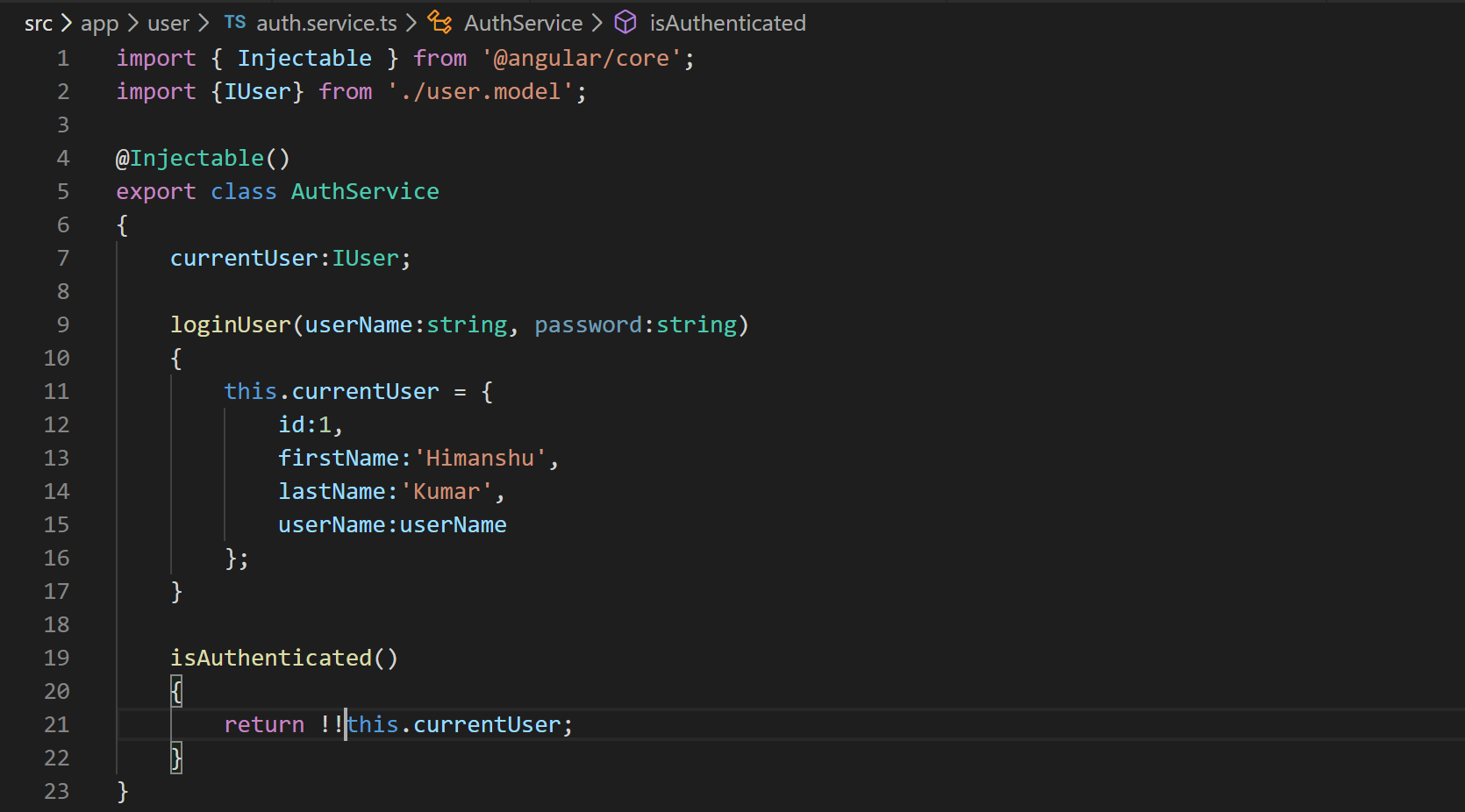
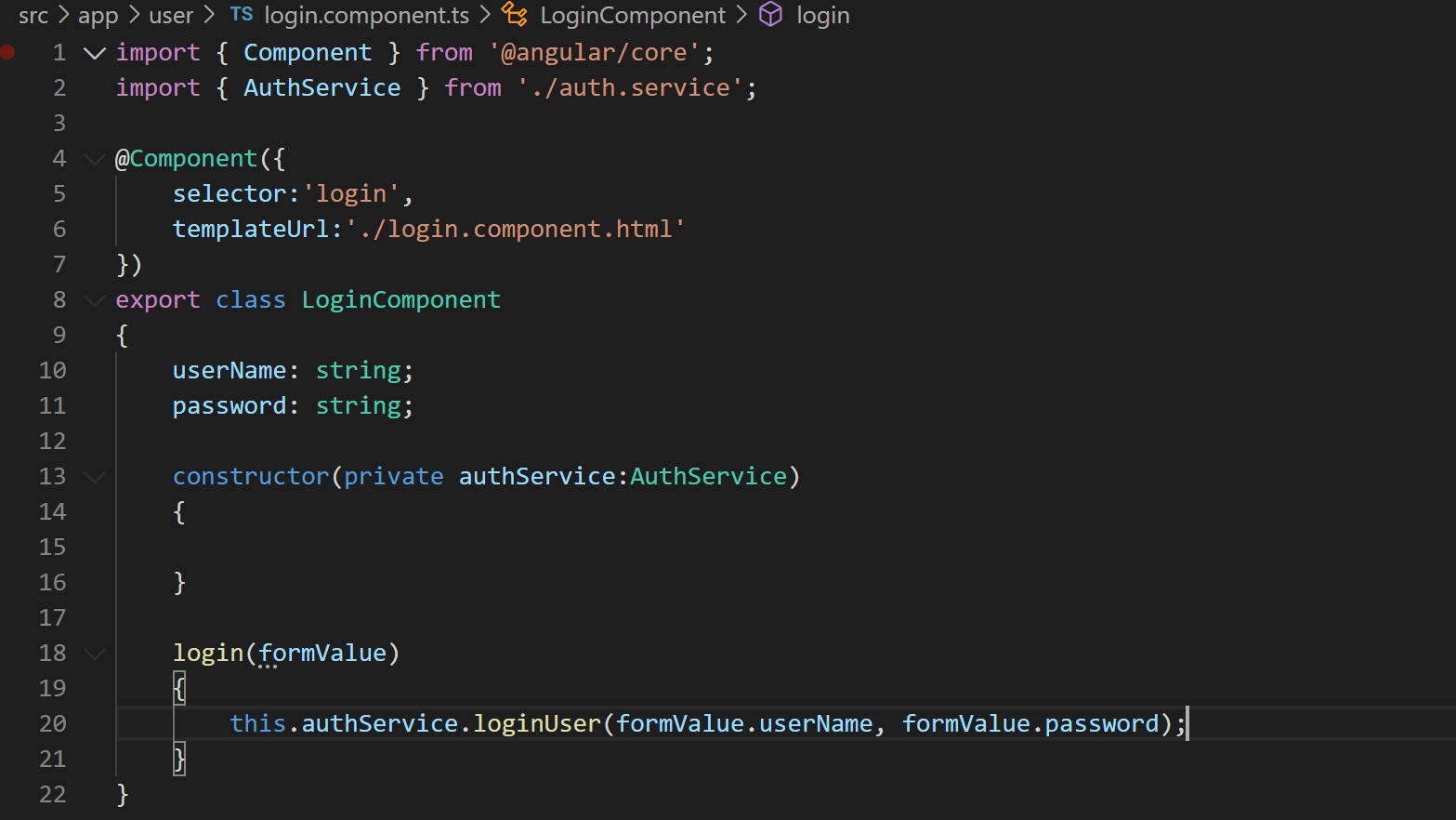
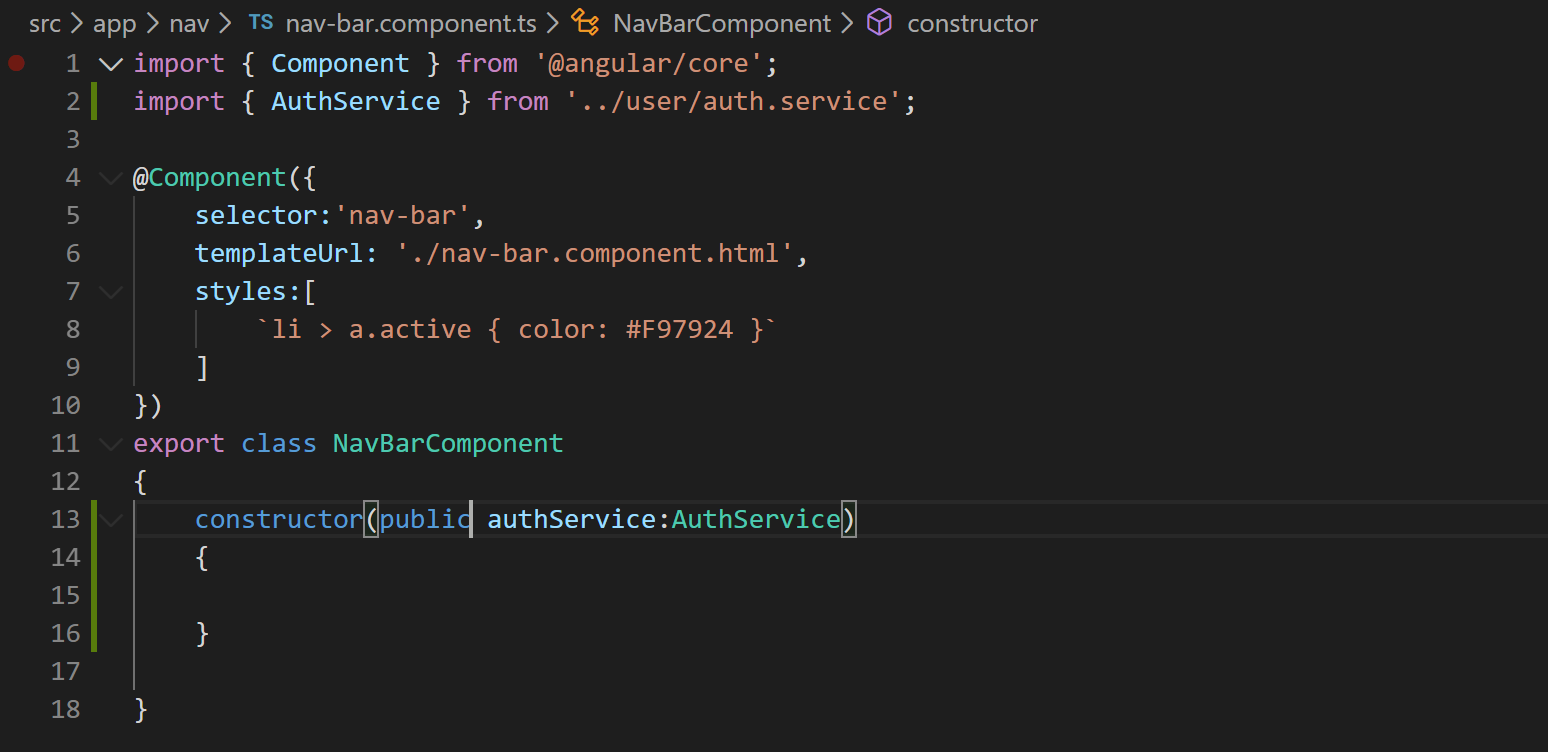
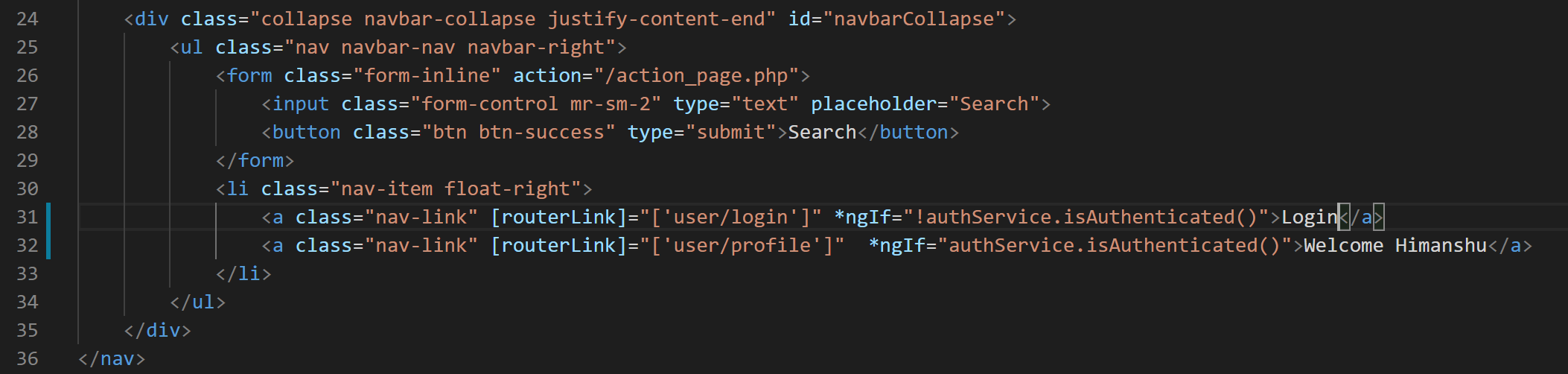
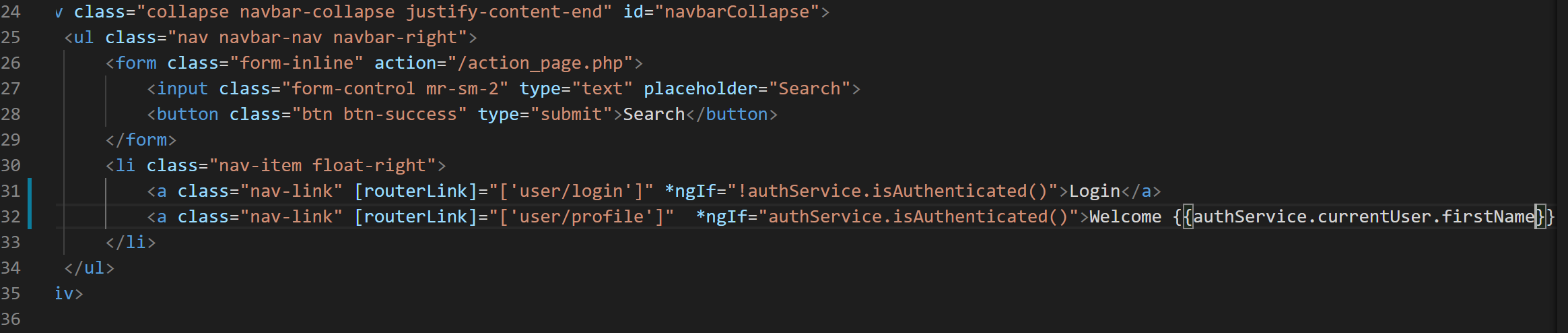
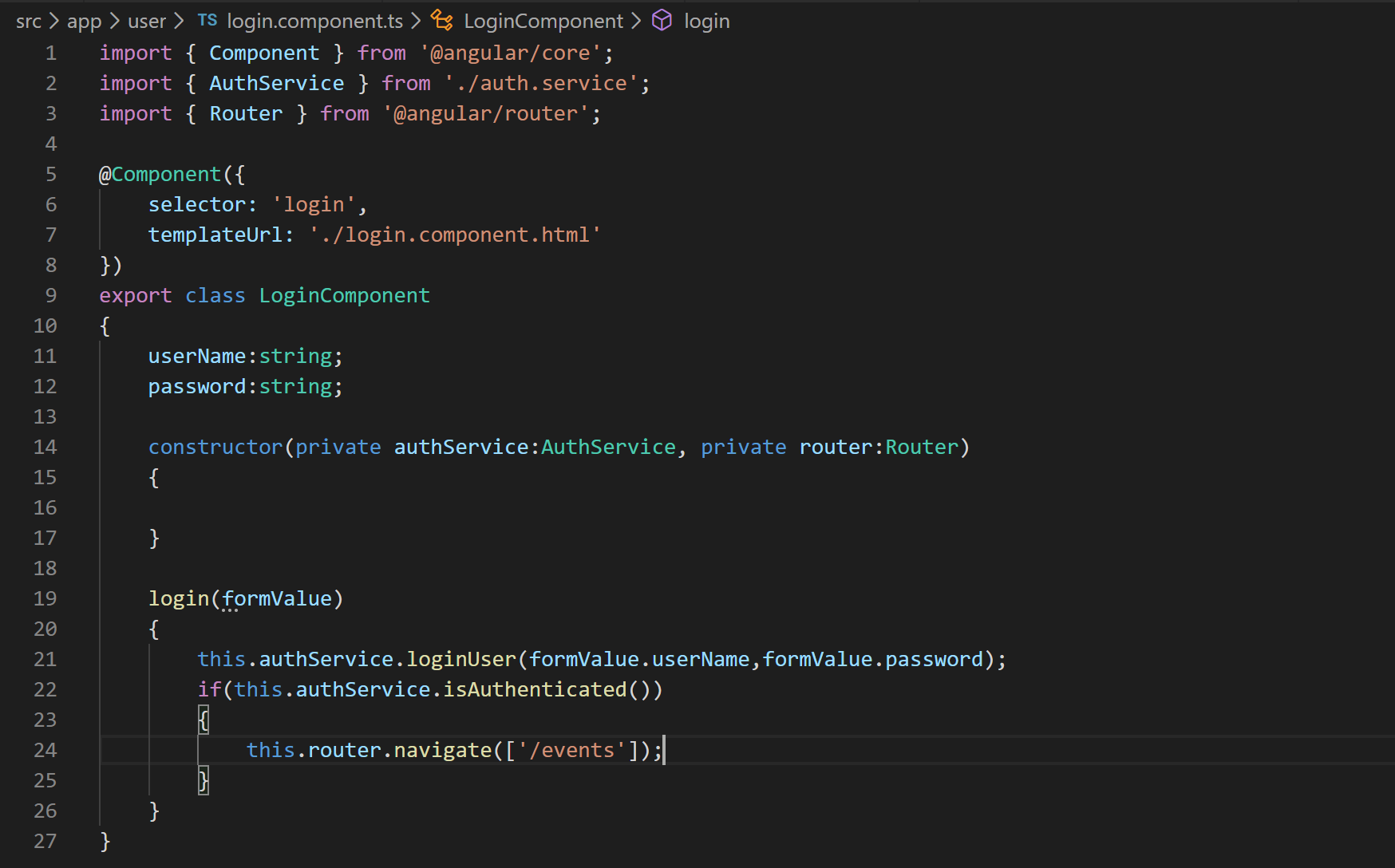
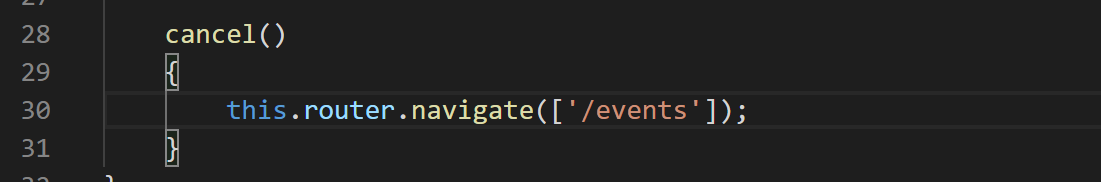
1. There are two ways you can create forms in Angular
   1. Template based forms – Good for small forms.
   2. Model based forms or reactive forms – Good for large forms since they allow unit testing
2. We will start with creating template based form.
3. Right click “user” folder and create a new file named as “login.component.ts”.
4. Add basic component shell to this file like below:
5. 
6. Now lets create the corresponding template file. So right click “user” folder again and create a new file named “.login.component.html”.
7. Lets copy and paste the HTML for this template from “misc” folder of cloned or downloaded Github repo.
8. The HTML that we just copied is just plain HTML there is nothing Angular specific in it yet. Now lets start adding Angular specific code to enable validations.
9. Before we start adding validation related code lets add this component in our user module and add its route in route config like below:
10. 
11. If you try to browse your login form by entering following URL in the browser “<http://localhost:4200/user/login>”. You should see the application like below:
12. 
13. Since we are now going to use validation related directives we will have to import these directives. The directives are defined in FormsModule. So lets import FormsModule in user module like below:
14. 
15. Do not forget to add the corresponding import statement for FormsModule. VS Code surprisingly did not add it automatically. “import { FormsModule } from ‘@angular/forms’”.
16. Alright lets add data binding statements to our template (login.component.html) to bind to component properties like below:
17. 
18. Ok now lets add “name” HTML attribute as one way binding requires this to be defined like below:
19. 
20. Now let us create form level local variable which will be used to pass form data to our component’s method like below:
21. 
22. We are now ready to pass this form data to component’s login method using “ngSubmit“ directive like below:
23. 
24. Lets now define the properties in our component. So open “login.component.ts” and define the properties like below:
25. 
26. If you notice VS Code terminal it is still complaining that login method is not there in the component. So lets define the method too like below:
27. 
28. Since we are console logging the form value lets take a look at what gets logged in the console when we click the login button on login form. So go to your application in the browser and open Developer tool bar.
29. 
30. It is logging the form object like a JSON as shown in above image. So good we are now getting the form values from the form to our controller ready to be consumed.
31. Ok lets consume the form values using an authentication service which will log the user in. So right click the “user” folder and create a file called “auth.service.ts”.
32. Create the basic service shell which you should be familiar by now how to create.
33. 
34. The service will have a loginUser method and a currentUser property of type IUser. So lets create an IUser model, loginUser method and currentUser property like below:
35. 
36. Create the IUser model in a file called “user.model.ts” in “user” folder and add the necessary import statement in AuthService like below:
37. 
38. 
39. We will not really authenticate the user as of now and fake it by setting the currentUser property in auth service like below:
40. 
41. In our code we will need to check if current user is authenticated or not. To check that lets add a method “isAuthenticated” to our service like below:
42. 
43. Our auth service is now ready to be consumed so lets add it as provider in app module. We are adding this service as provider in App module because it will be used by other components also about the login status of current user.
44. 
45. Now lets inject the auth service in our login component and use it login the user like below:
46. 
47. Now that we have wired up the login method in the component. Lets show a login link in the navigation bar when the user is not logged in and welcome message when the user is logged in. So lets open the “nav-bar.component.ts” and inject the auth service like below:
48. 
49. Now that we have injected the auth service lets use it to render a Login link when user is not logged in and welcome message when user is logged in. We will use “ngIf” template directive to achieve that like below
50. 
51. Now lets update the welcome message with a binding to show the first name of currently logged in user.
52. 
53. Now that logging in is complete with service let us fix some small issues like redirecting to all events page when login is complete and when user clicks cancel button. So lets import Angular’s router service in “login.component.ts” and use it navigate to all events page like below:
54. 
55. Lets add the cancel method like below:
56. 
57. Lets wire up the “cancel” button in the template like below:
58. 