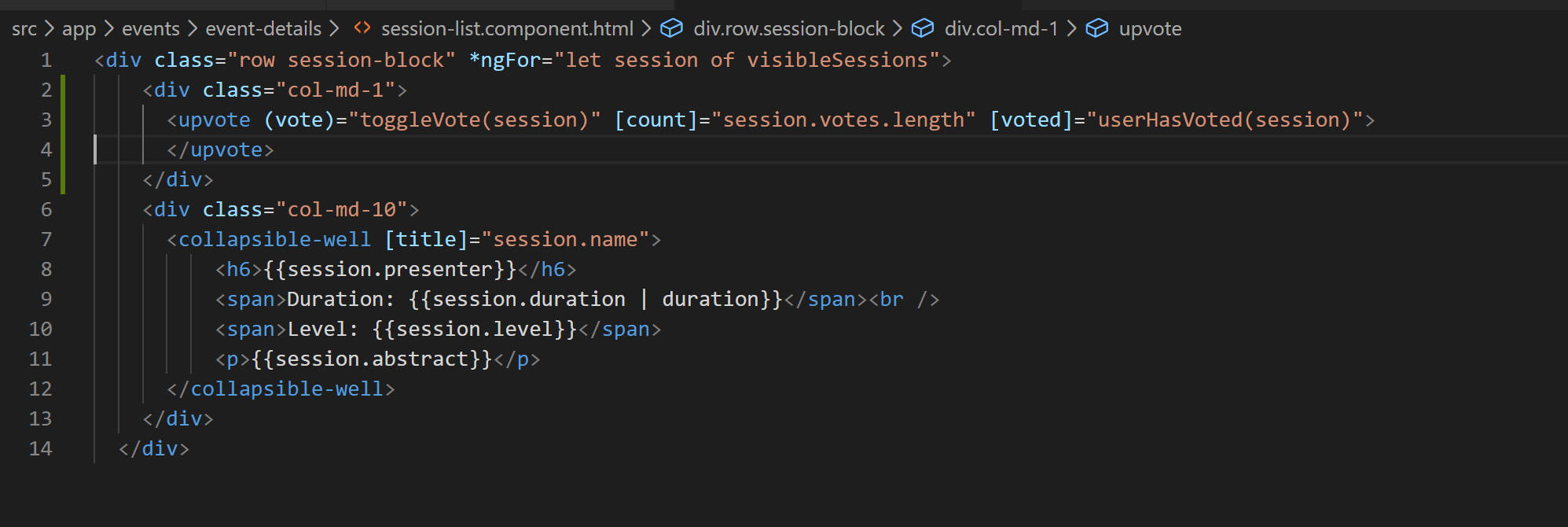
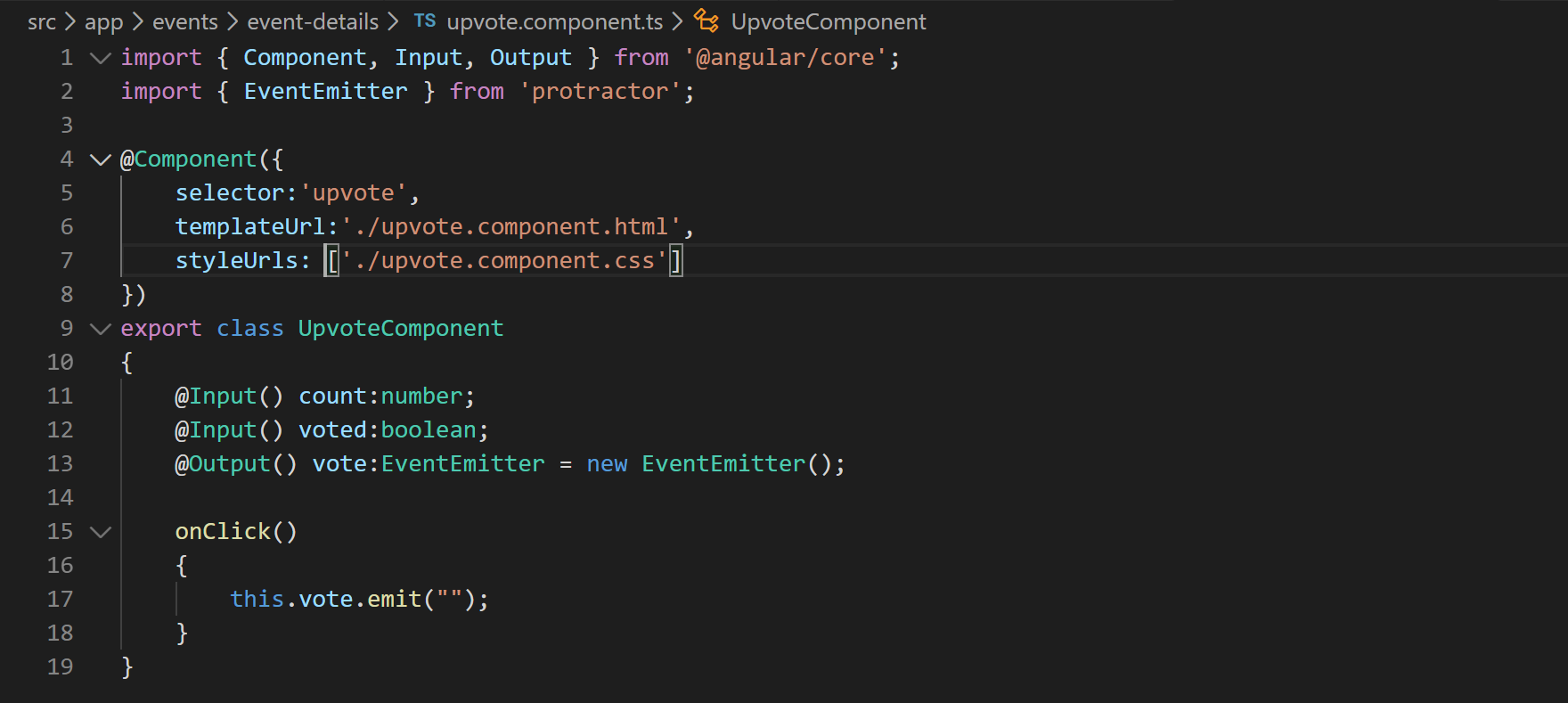
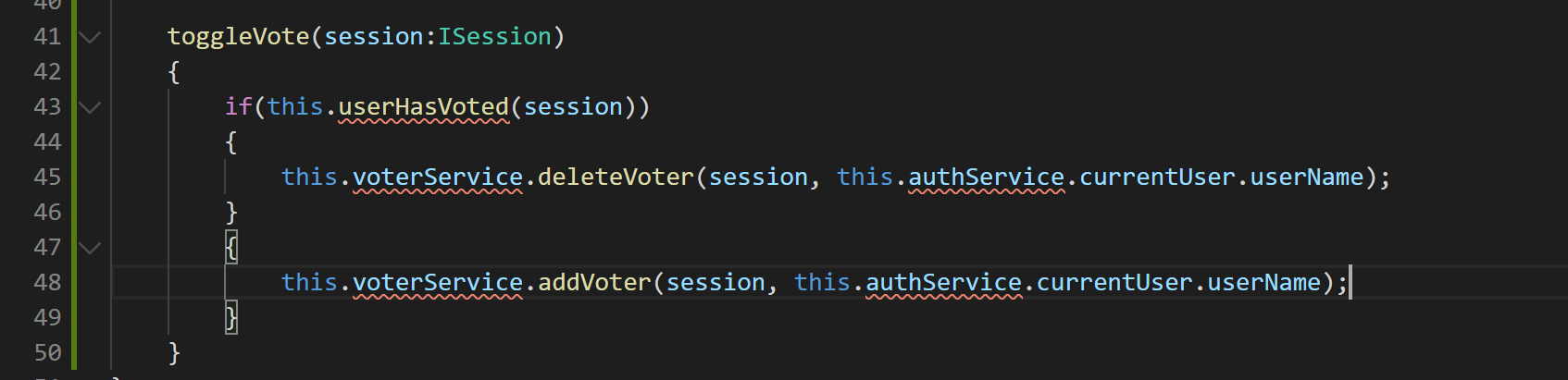
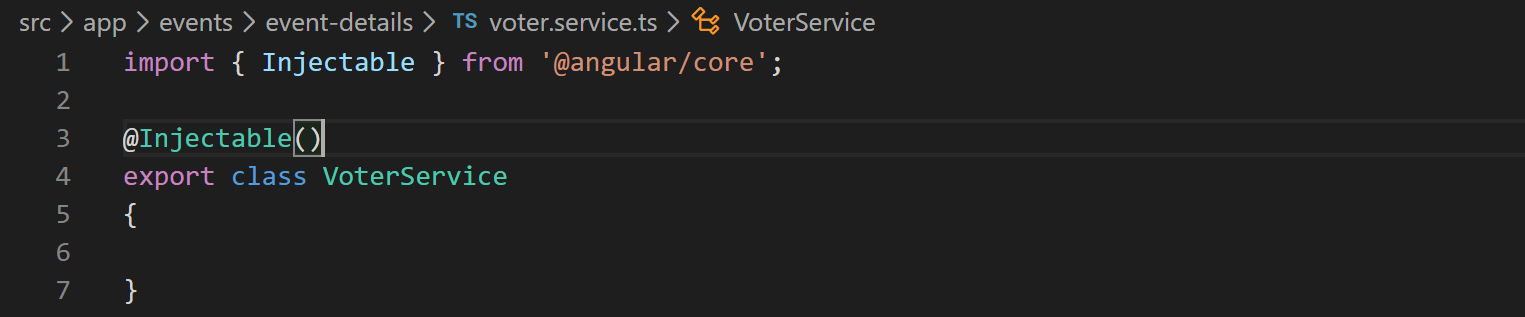
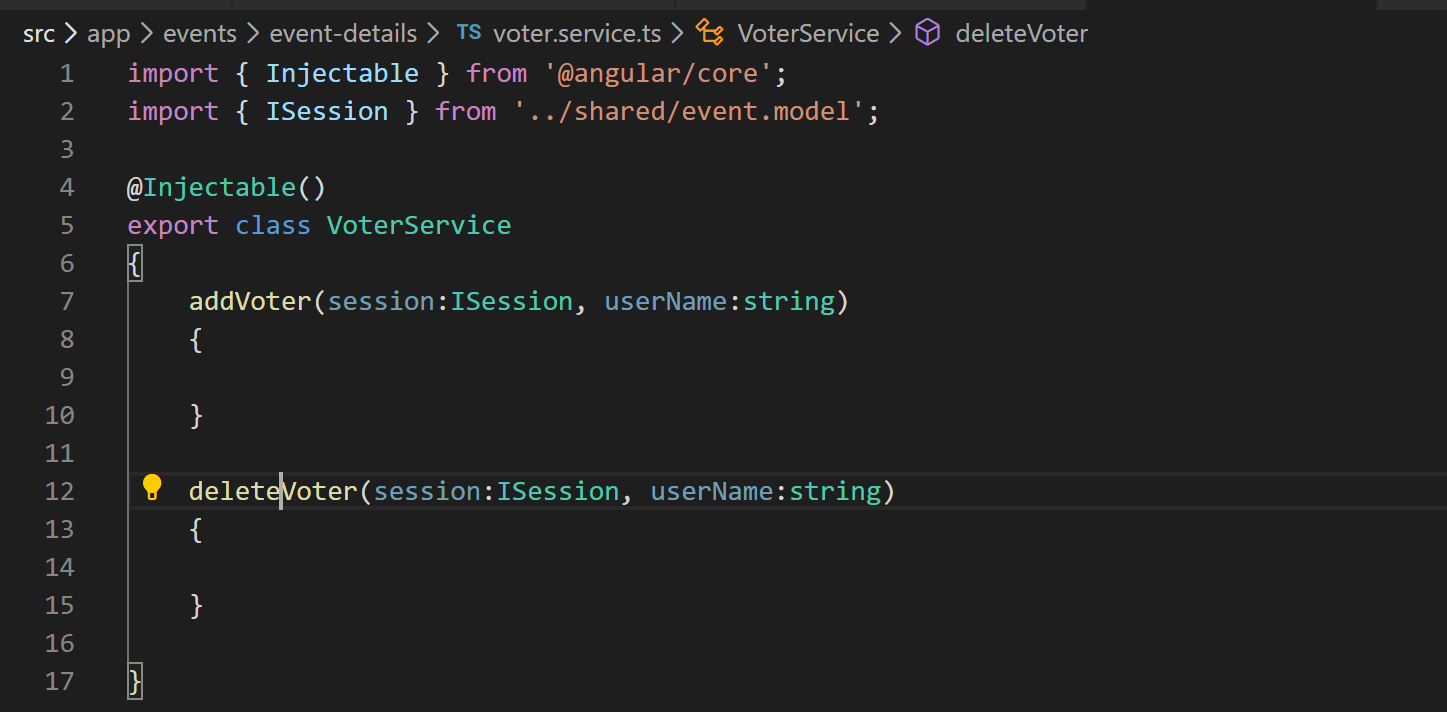
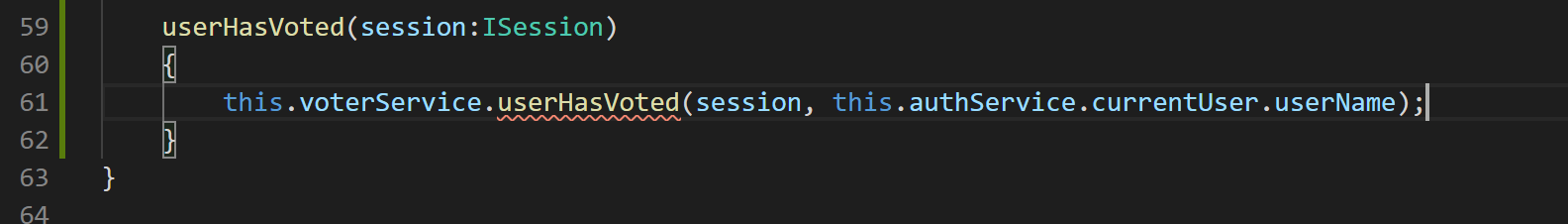
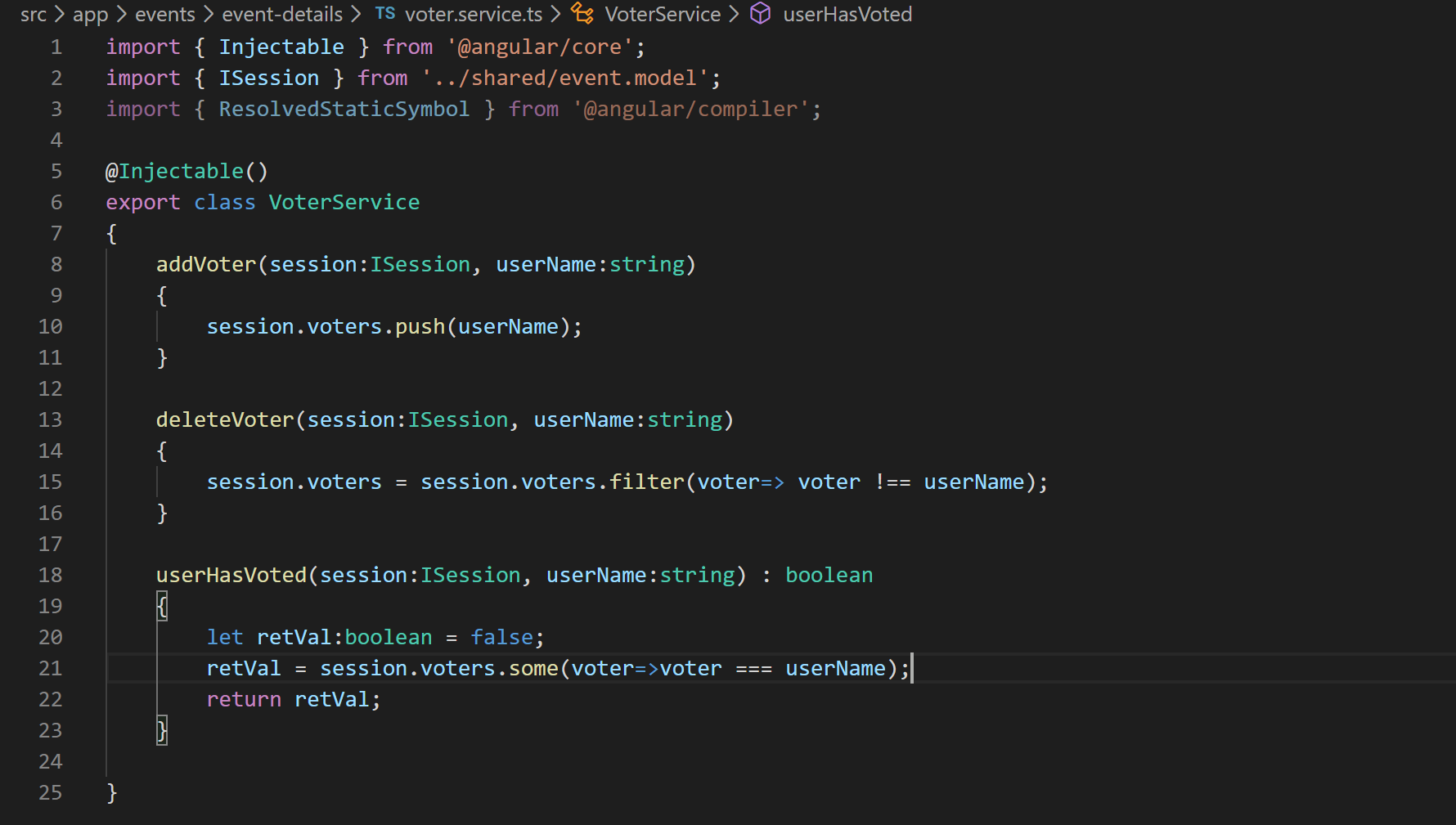
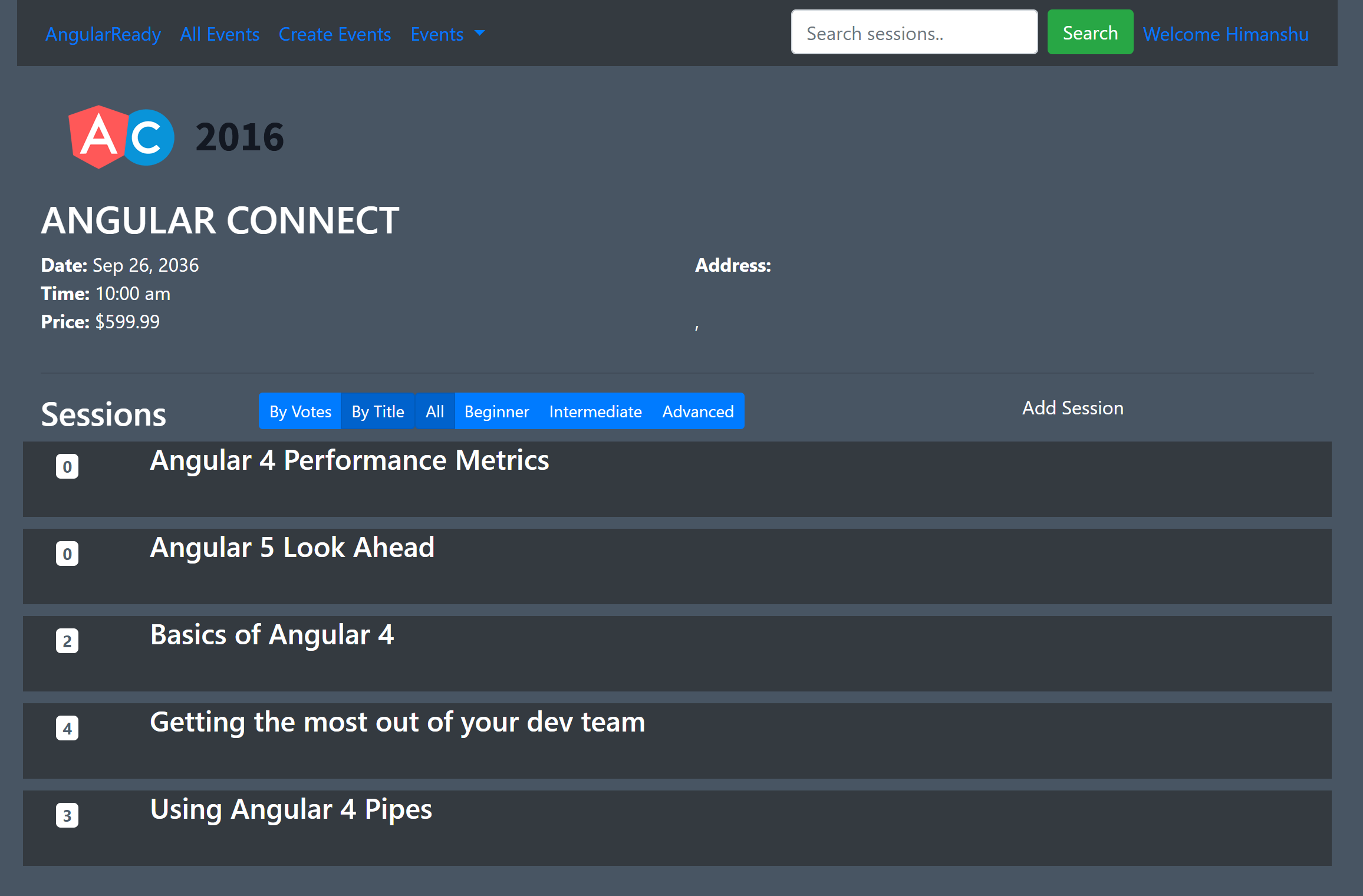
1. We will now implement voting functionality so that users can vote for sessions. So lets define the HTML, event handlers and properties for our voting component.
2. Open the file “session-list.component.html” and define the voting component’s HTML, event handlers and properties like below:
3. 
4. We have defined the following:
   1. A component named as “upvote”.
   2. A method handler named as “vote” and we are calling a function “toggleVote” on it.
   3. A property “count” which determines the total vote count.
   4. Another property called “voted” which determines if current user has voted or not. For that we are calling a function “userHasVoted”.
5. Lets create the file in “event-details” folder called “upvote.component.ts”.
6. Lets add the basic component shell and properties and methods like below:
7. 
8. Now lets define the HTML template file for this component in a file called “upvote.component.html”.
9. Lets grab the HTML from “misc” folder of cloned or downloaded repo.
10. Now lets register the component in app module’s declarations array.
11. As you can see in the HTML template we have defined quite a few custom styles for this component so lets define them in a separate CSS file.
12. So create a file called “upvote.component.css” in “event-details” folder.
13. Now lets refer to the file using “styleUrls” property in component decorator in “upvote.component.ts” file like below:
14. 
15. Now lets grab the CSS file from the “misc” folder of cloned or downloaded repo.
16. If you notice VS Code terminal you will see that it is erroring out and complaining about toggleVote method so lets start implementing that “session-list.component.ts. file.
17. So we need to let user vote if he hasn’t voted and unvote if has already. So lets implement the check like below:
18. 
19. Now as you can notice we have defined a new service called voterService which will encapsulate the code for voting and unvoting. And we need to call AuthService to get the current user object. So lets inject AuthService in this component.
20. Now lets go a define voterService in folder called “event-details”.
21. So create a file called “voter.service.ts”.
22. Add the basic shell to it like below:
23. 
24. Now lets add the service in providers array in app module.
25. Ok now lets inject the service in session-list component.
26. Now we can add the addVoter and deleteVoter method skeletons in the service like below:
27. 
28. Alright now lets implement the method “userHasVoted” in “session-list.component.ts” like below:
29. 
30. So now the voterService does not have userHasVoted method so lets add that in our service.
31. Alright now lets implement the methods in voter service like below:
32. 
33. Ok with now lets try to browse the application and see if we are getting to see votes and icons or not.
34. You will have to login to use this feature.
35. 
36. As you can notice the application renders fine but the voting button is not there. That’s because we used “glyphicon” which bootstrap 4 does not support. So lets install font-awesome and use those icons which bootstrap supports.
37. So in VS Code terminal press Ctrl+C and run following command:
38. “npm install --save font-awesome”.
39. Now lets add the font-awesome css in styles section of angular.json file like below:
40. 
41. Now start the server again using the command “ng serve”.
42. Ok now open the file “upvote-component.html” and replace “glyphicon” class with “fa”.
43. Replace “glyphicon-heart” with “fa-heart”.
44. Replace “glyphicon-heart-empty” with “fa-heart-o”.
45. Now if you render the application it should look like below:
46. 