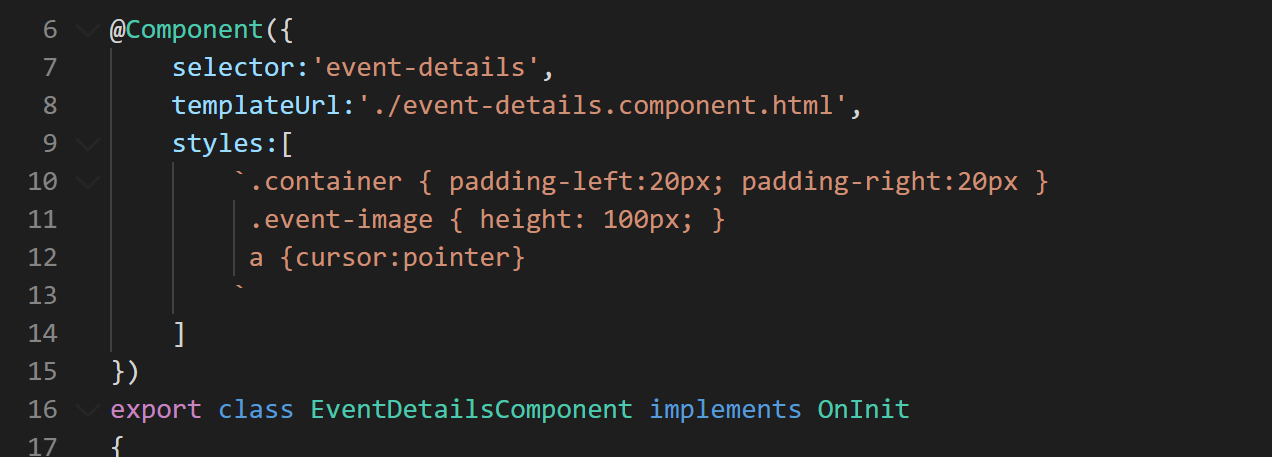
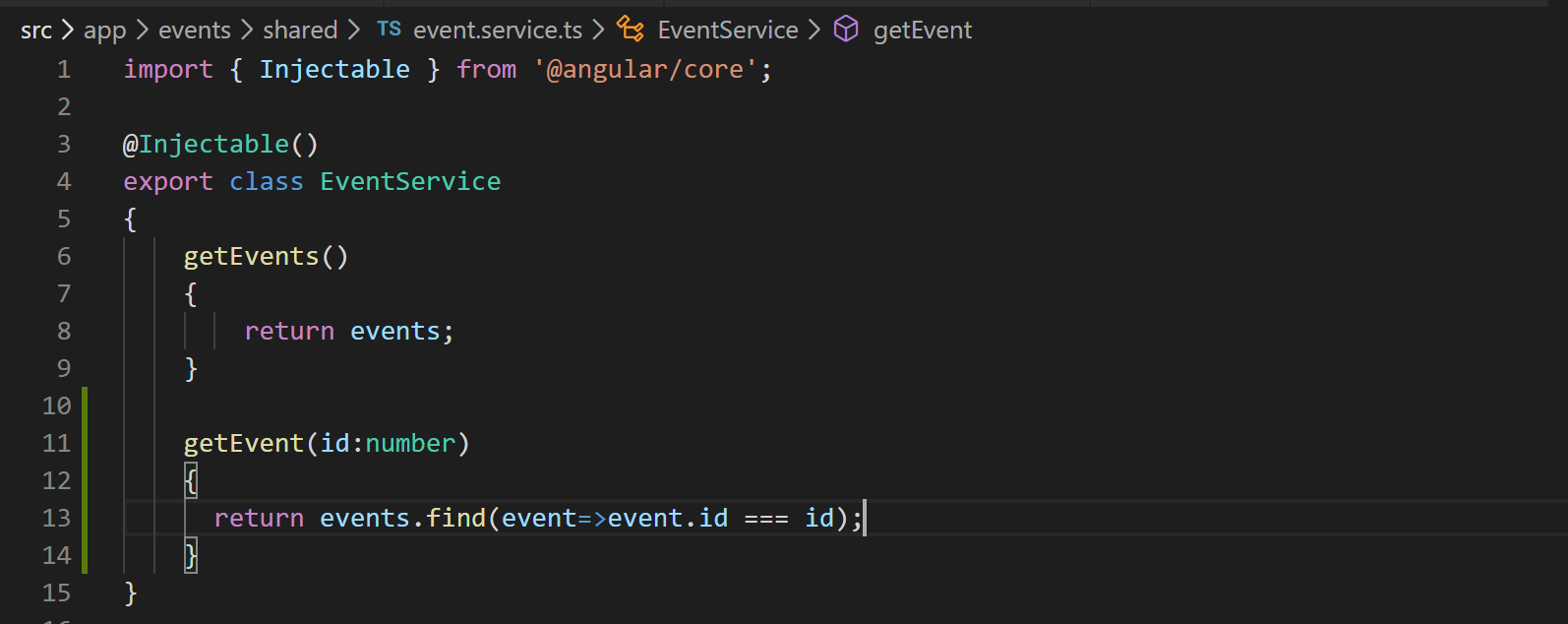
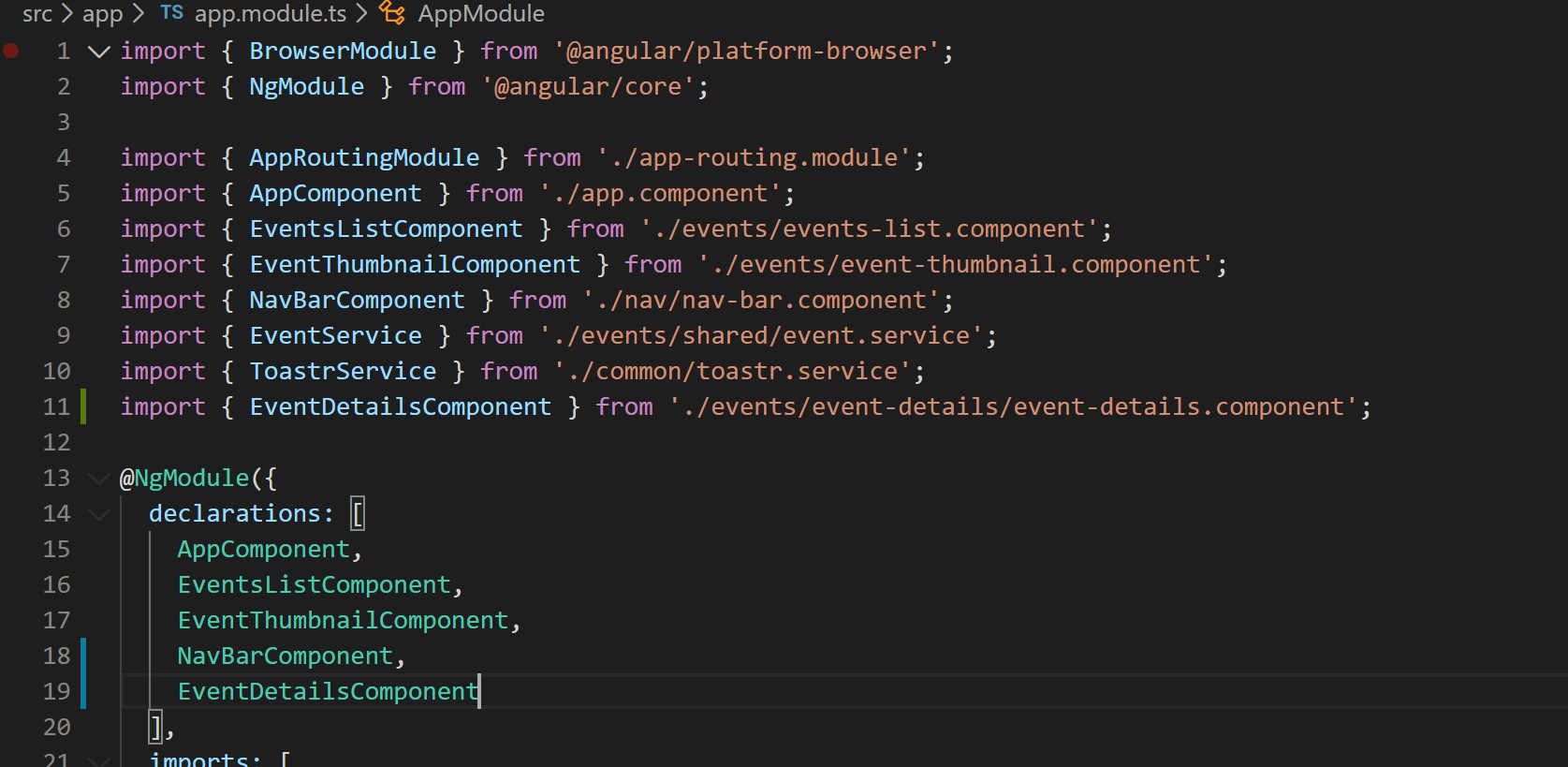
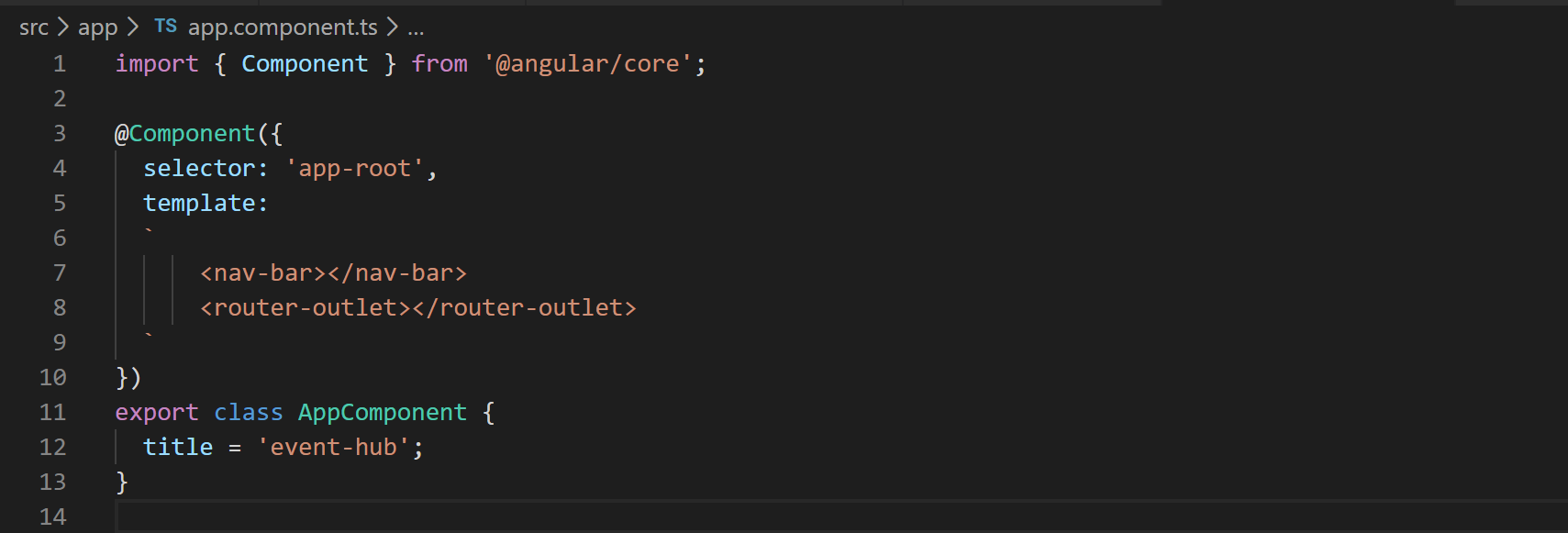
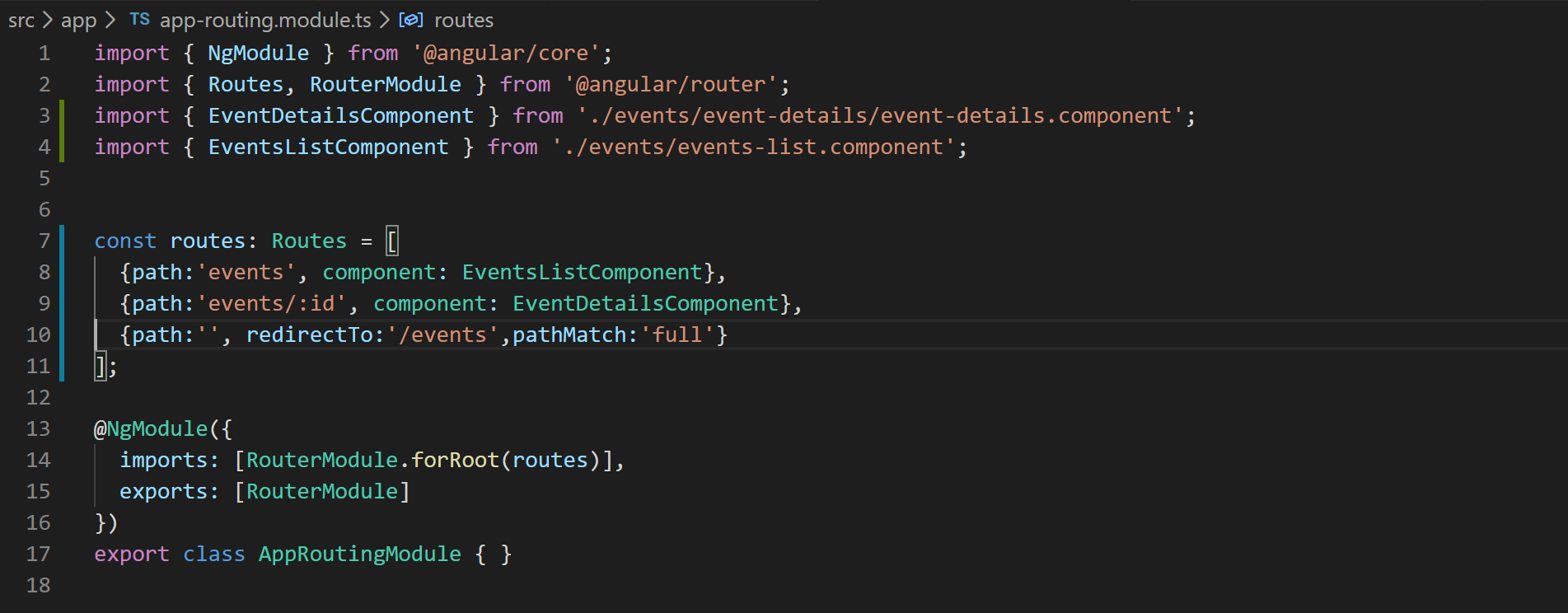
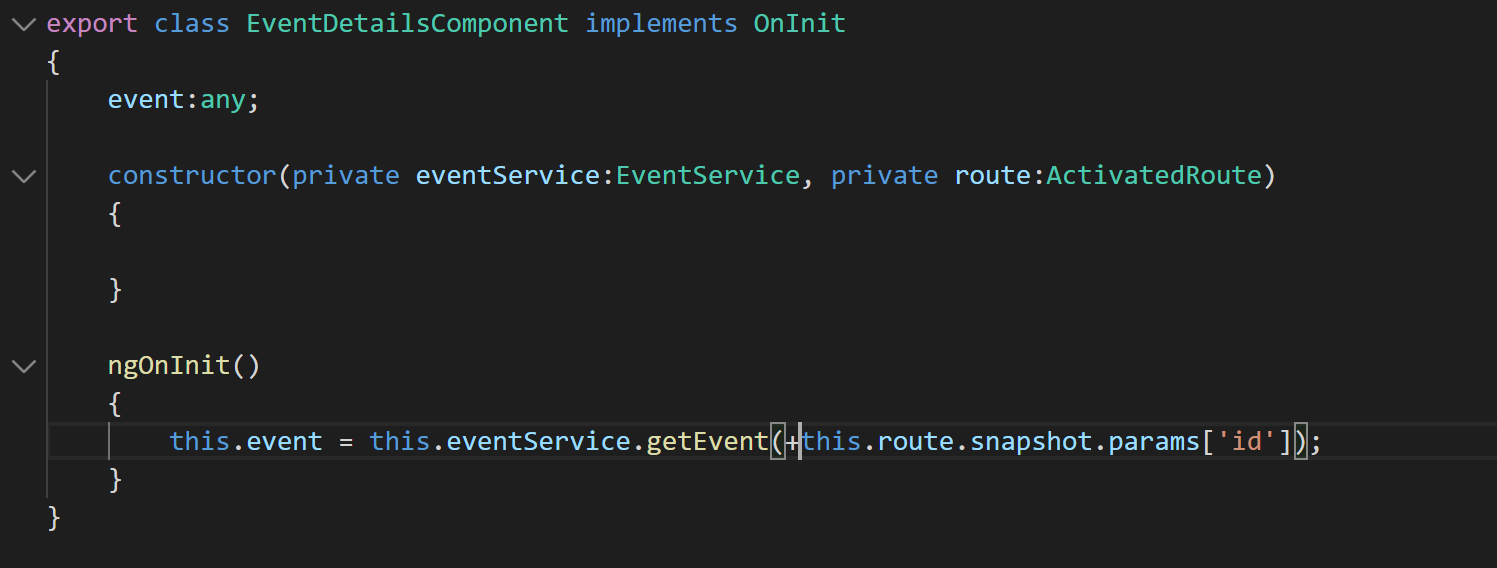
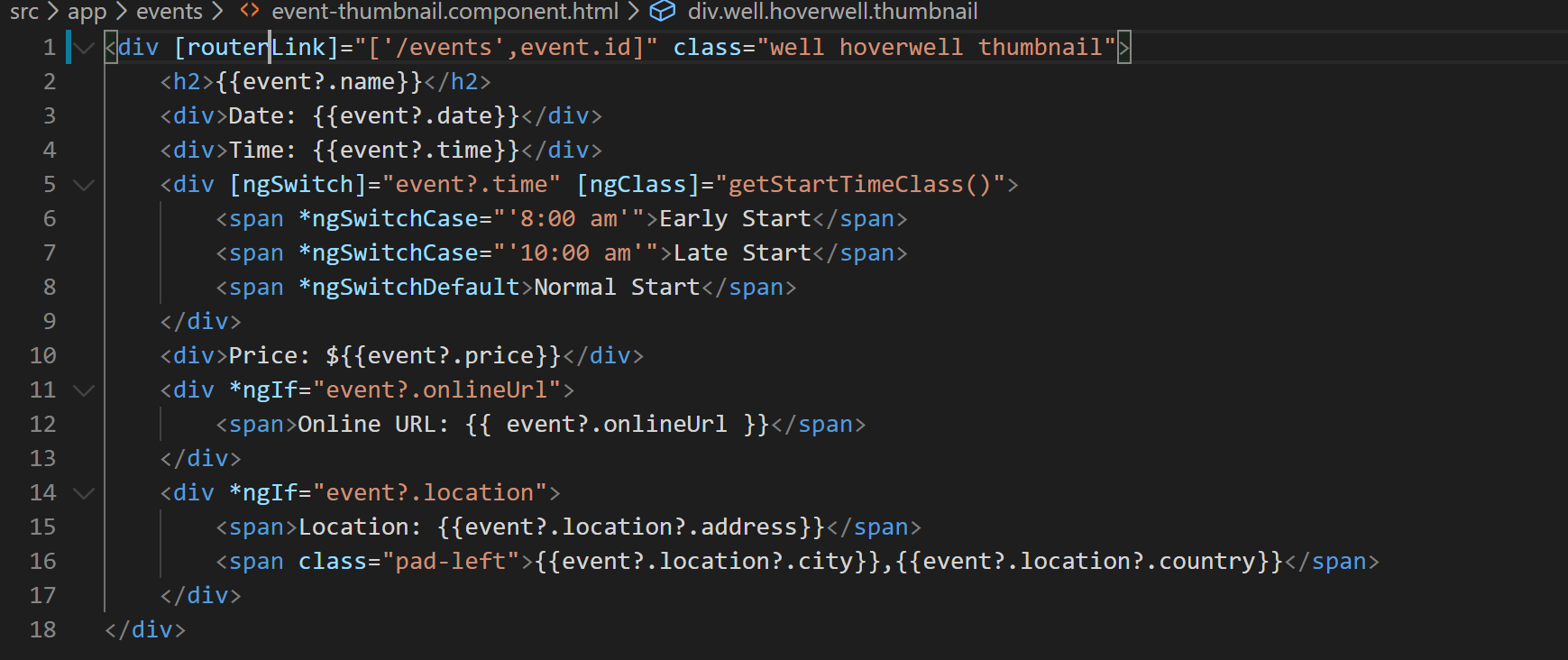
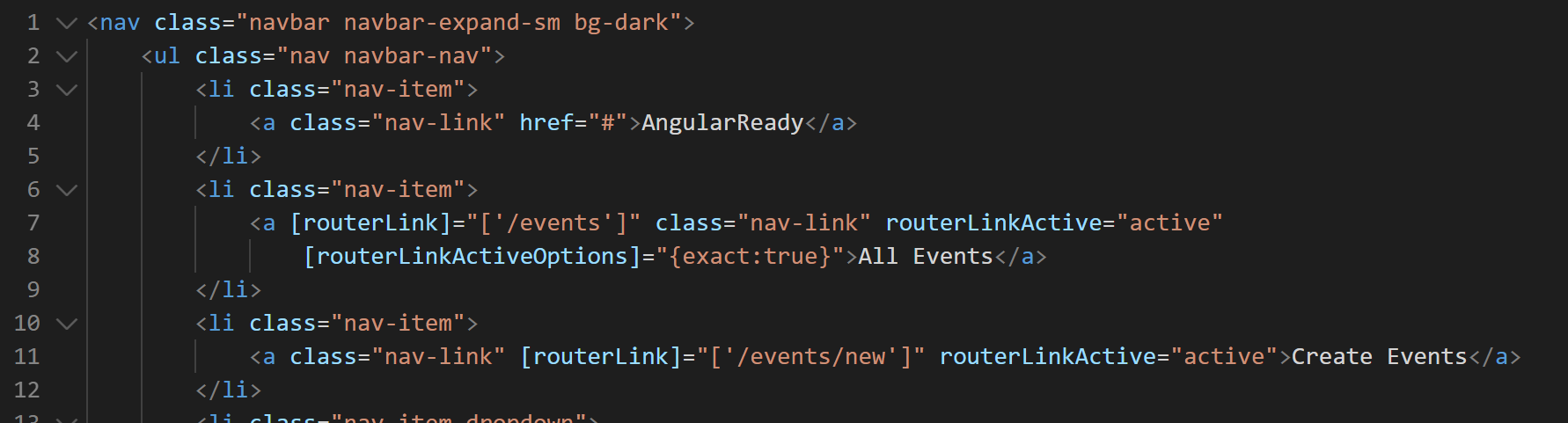
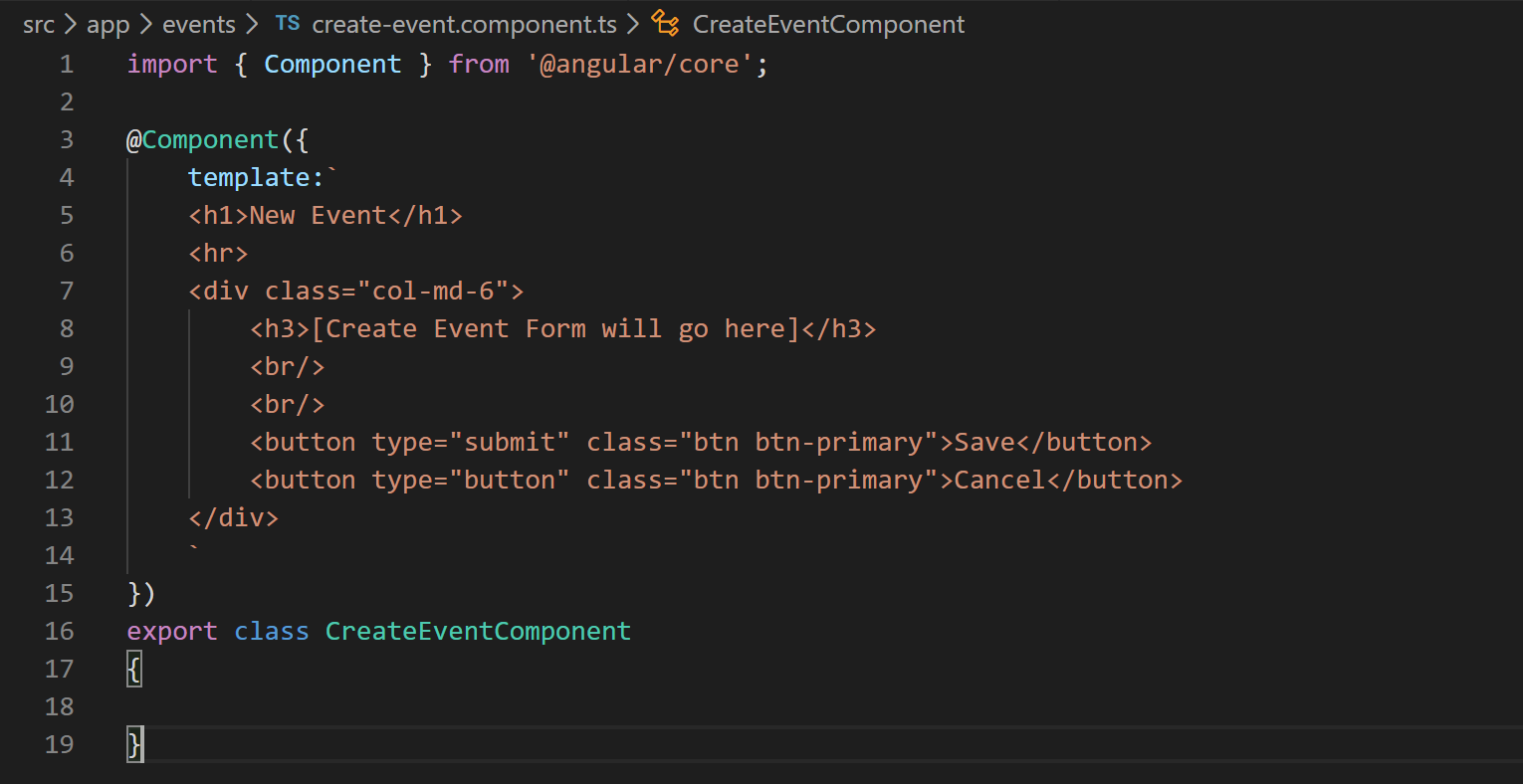
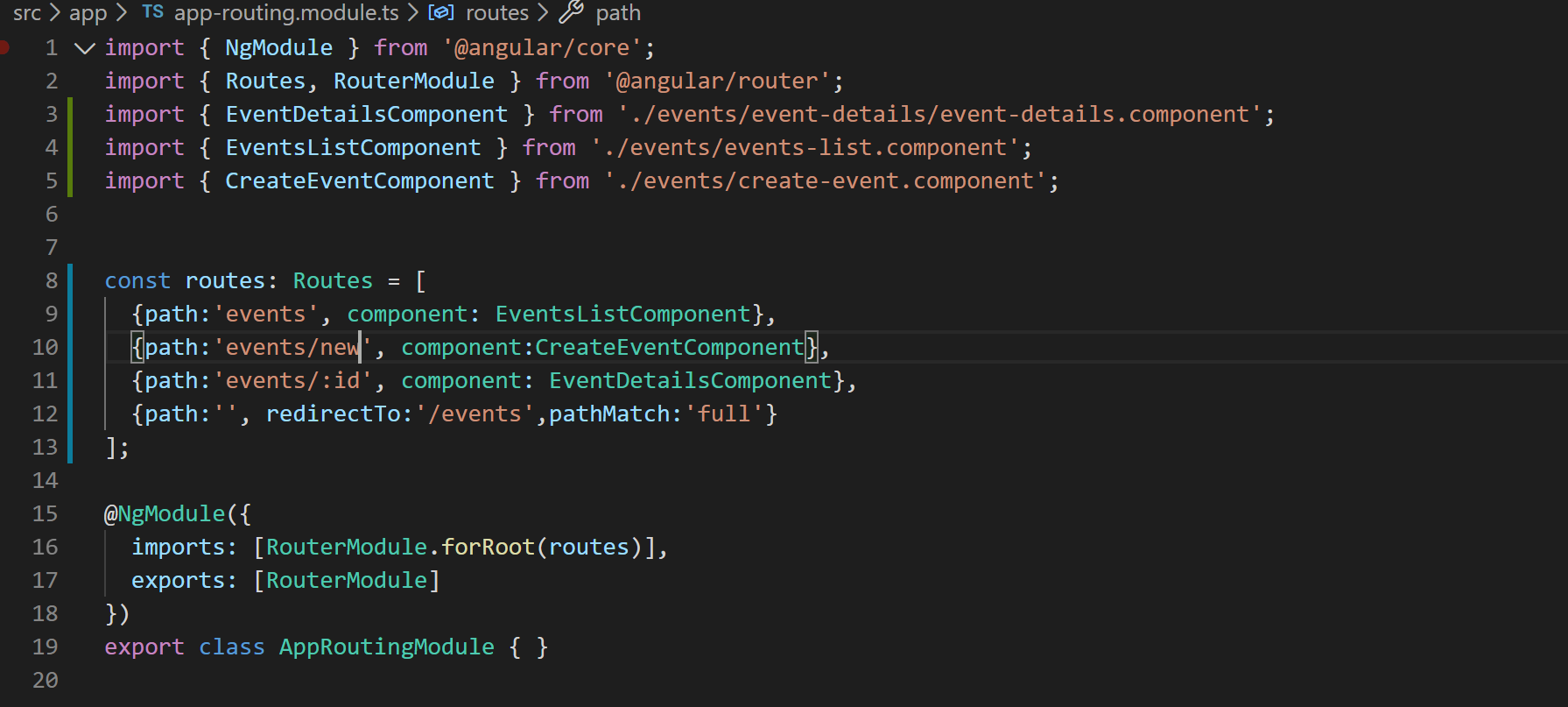
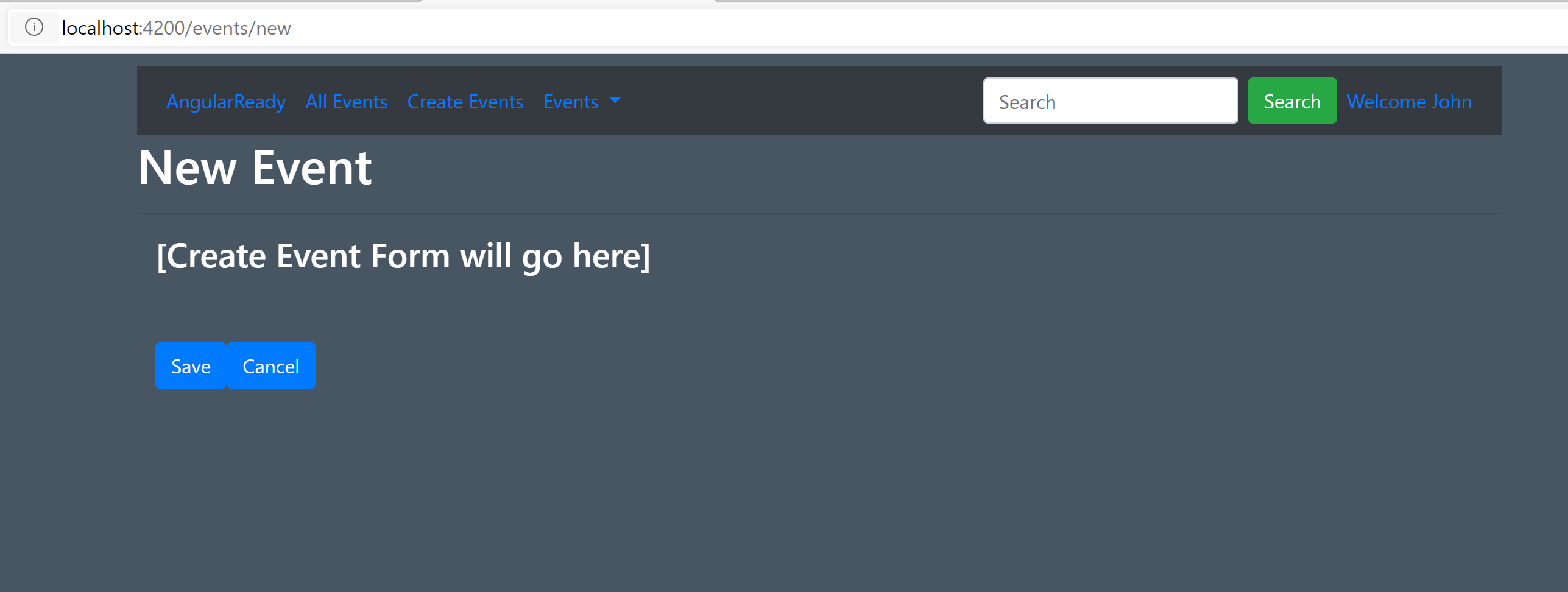
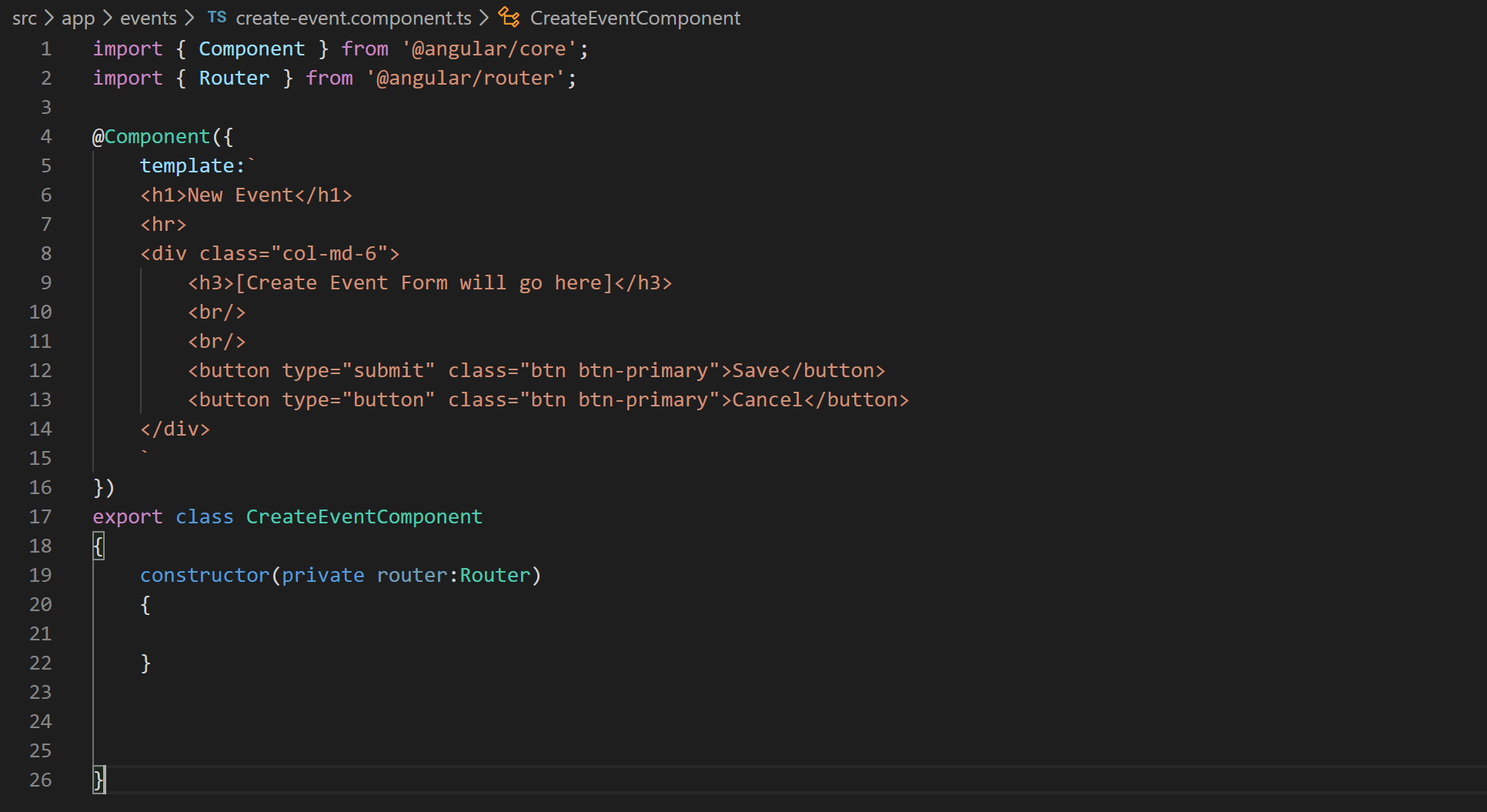
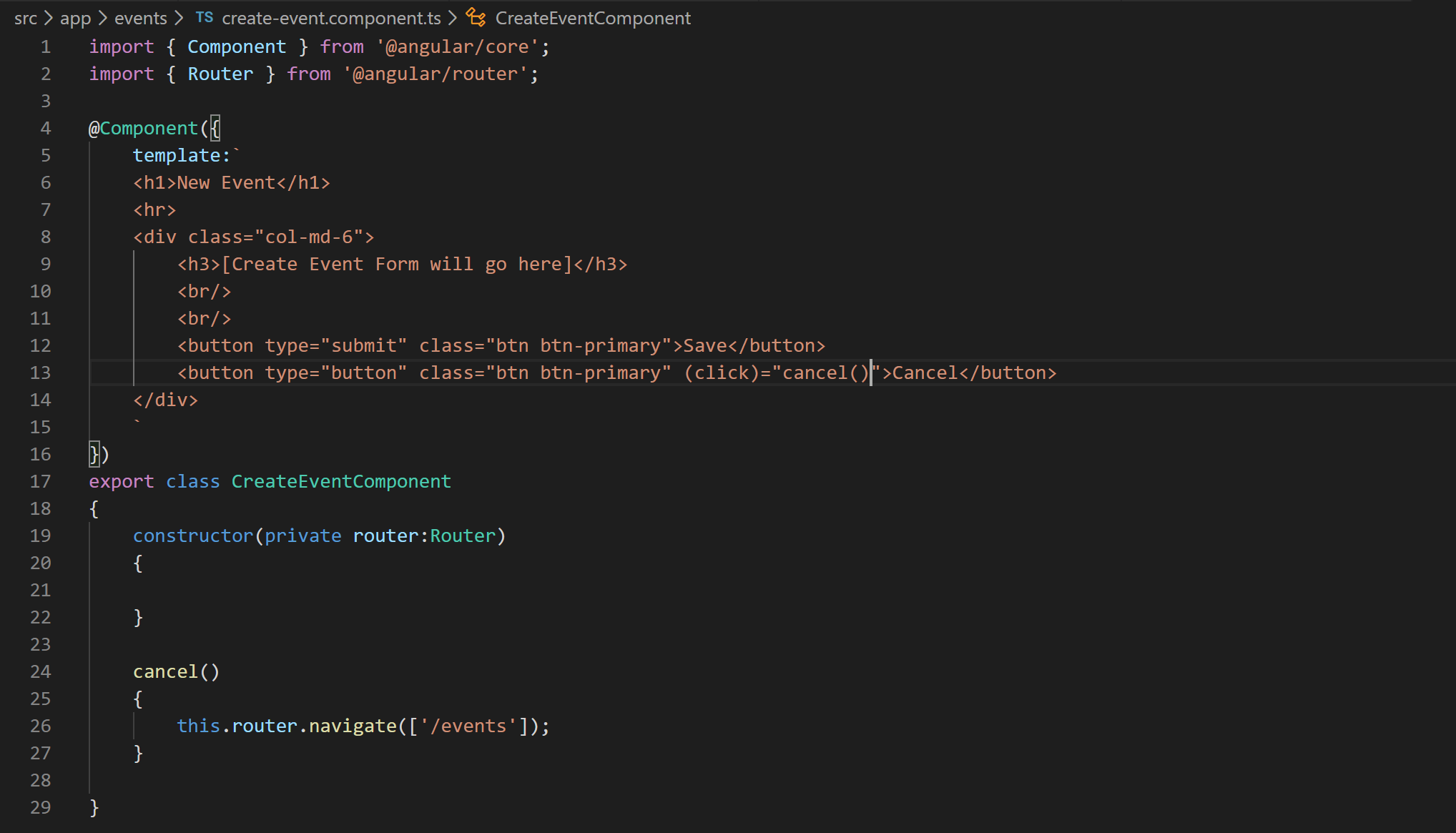
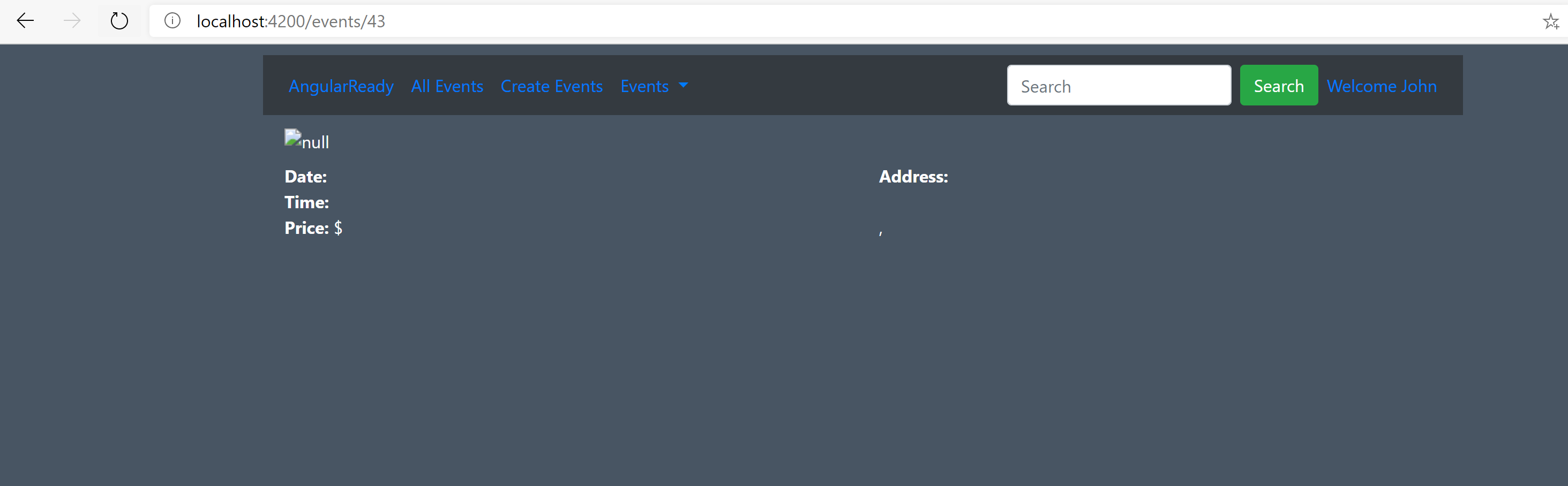
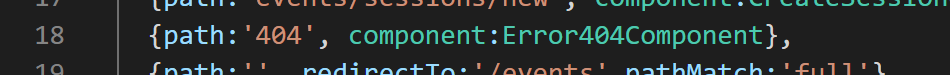
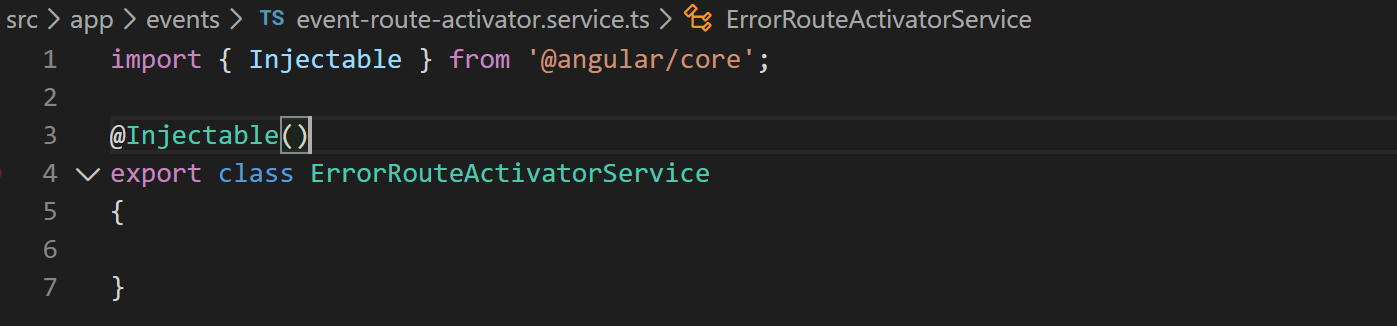
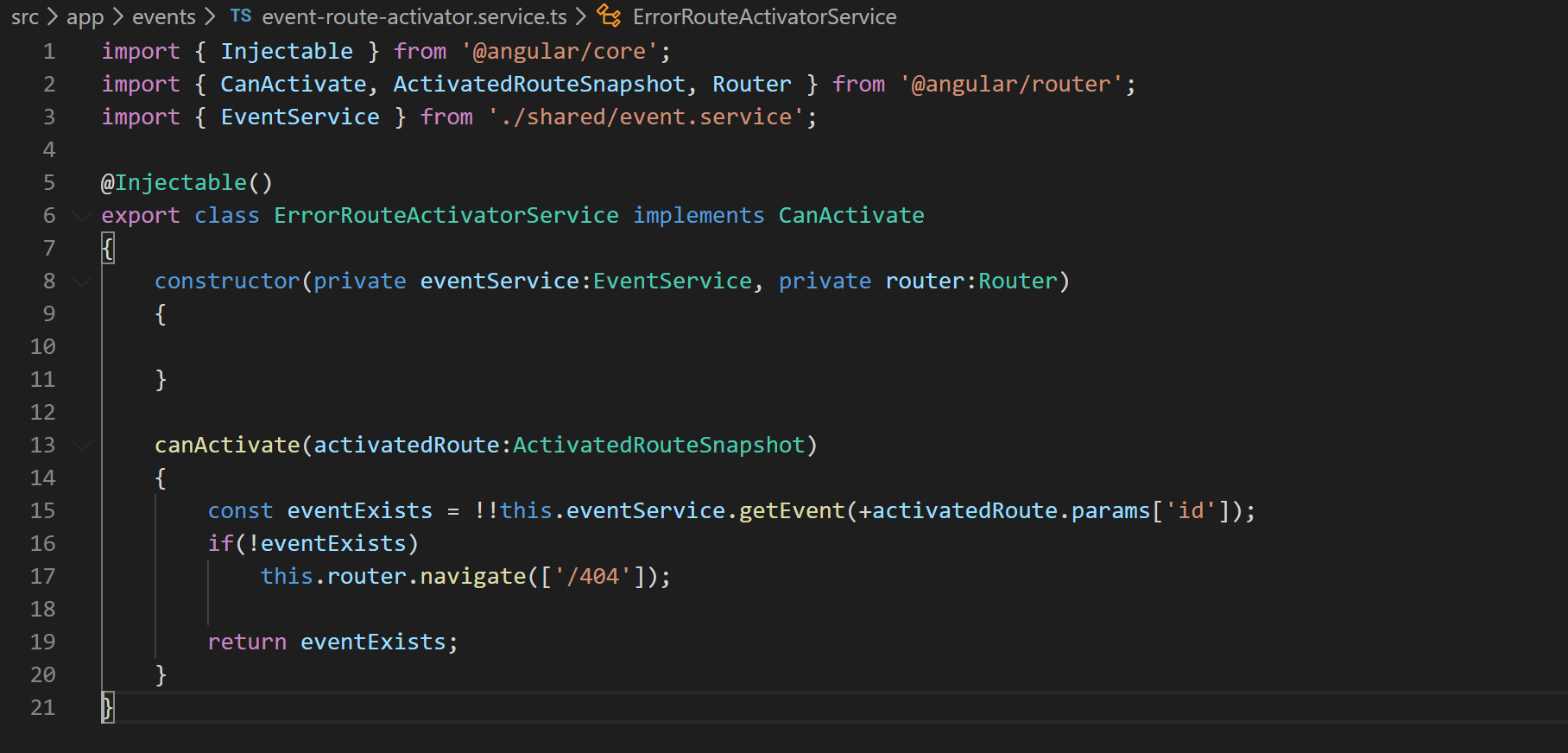
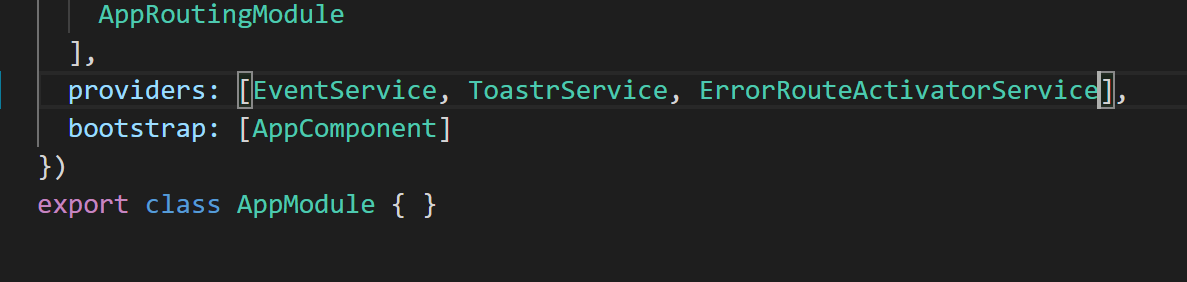
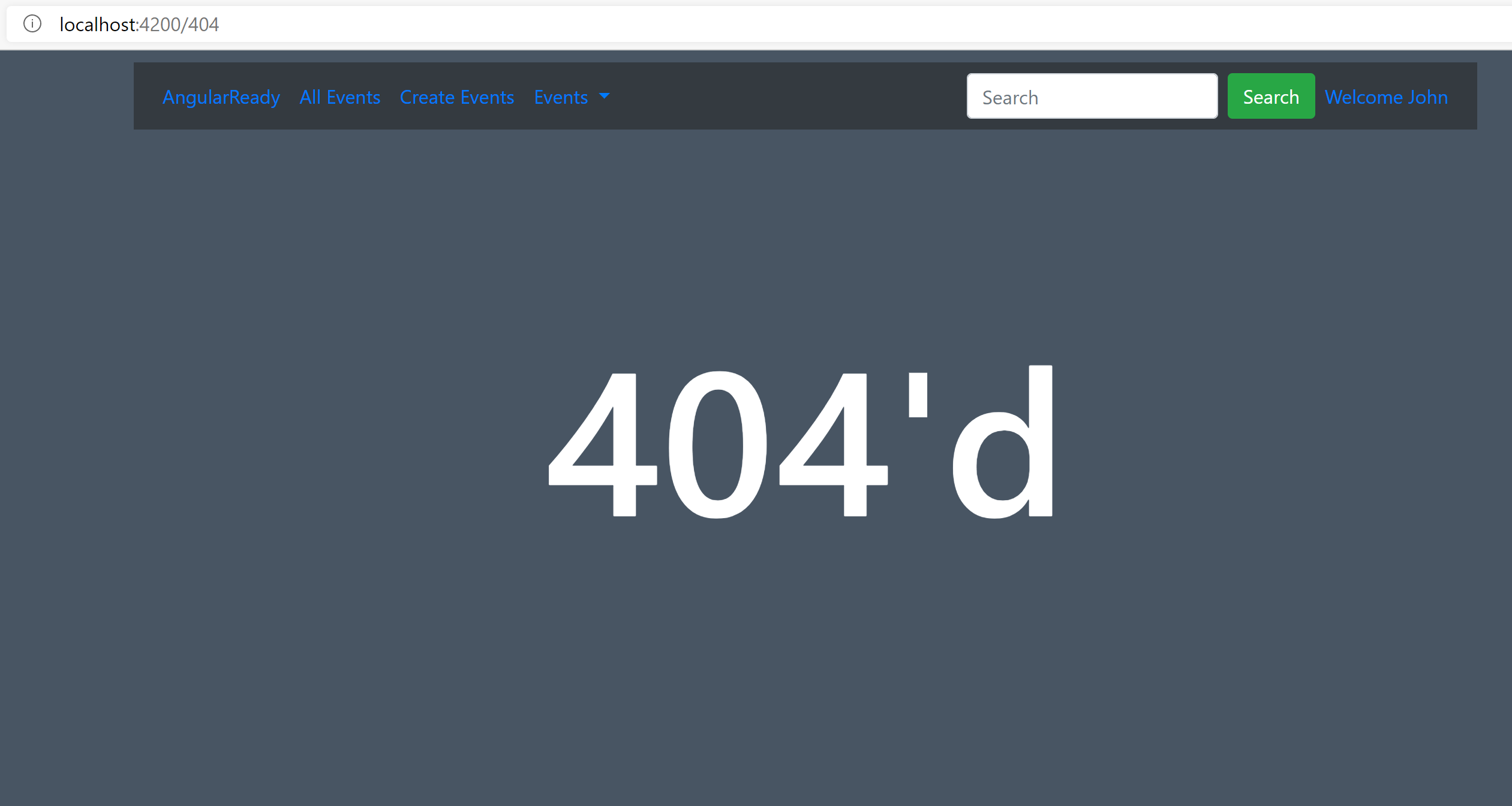
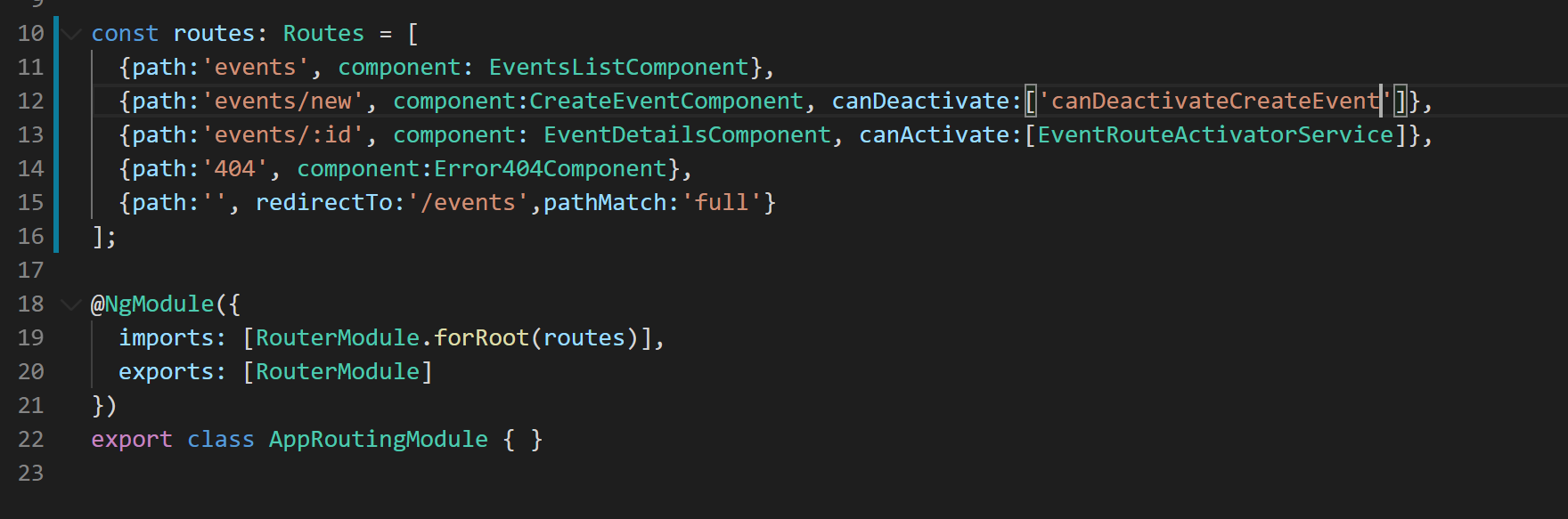
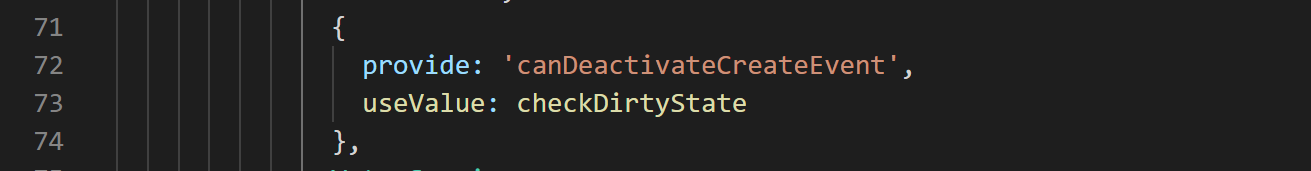
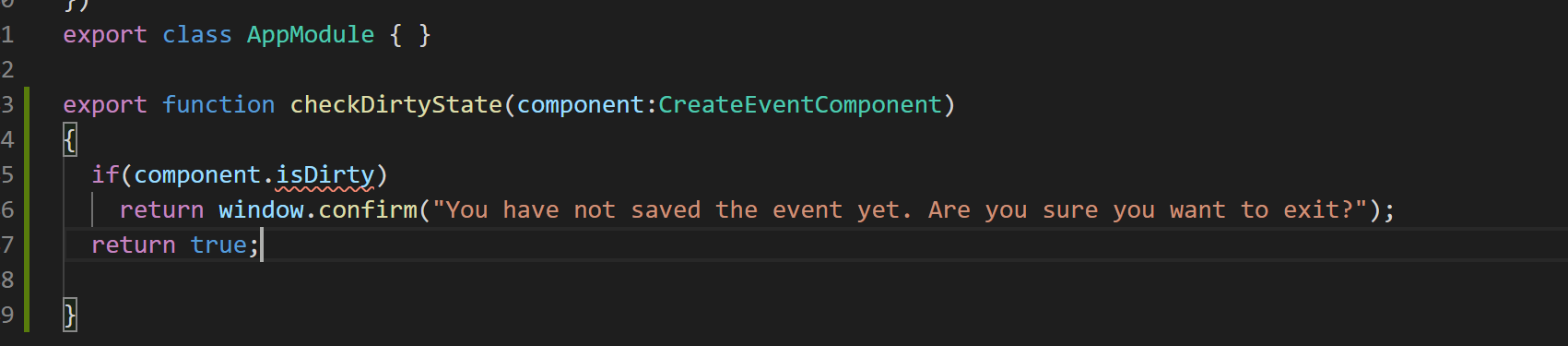
1. Right click “events” folder and click on “New Folder”. Name the new folder “event-details”.
2. Right click “event-details” folder and click on “New File”. Name the new file “event-details.component.ts”.
3. In “event-details.component.ts” write basic skeleton for the event details component like below:
4. 
5. Now let's create the event details HTML template. Right click on “event-details” folder and click “New File”. Name the new file “event-details.component.html”.
6. Copy the HTML for this template from “misc” folder in the cloned or downloaded repository. Paste the copied HTML so your file looks like below:
7. 
8. Now lets add some styles for event details component in the file “event-details.component.ts” like below:
9. 
10. Now lets apply these style to event-details HTML template like below:
11. 
12. Since the event details page is going to render details of one event we need some way of fetching the event by its ID. So let's add a get event method in our event service that will return event by matching its ID.
13. Open the file “event.service.ts” and add a method like below:
14. 
15. Now let's call this method in the ngOnInit event lifecycle hook in our event details component like below:
16. 
17. This component is now ready for use now let's register it in the app module like below:
18. 
19. Now lets add <router-outlet> component to our app component like below:
20. 
21. Now lets define the routes in our “app-routing.module.ts” file. Every route in the route array is an instance of a route object. Refer to the image below to define the routes.
22. 
23. Now if you hit the URL <http://localhost:4200> you will be automatically be redirected to <http://localhost:4200/events>.
24. If you hit the URL <http://localhost:4200/events/1> it will load the details of event with ID 1 and show the event details component.
25. Now let's parameterize the event-details component. We are already passing the event ID as parameter in route. Lets try fetch it. To fetch it we will have to use a class called “ActivatedRoute” like below:
26. 
27. Now if you hit the URL <http://localhost:4200/events/2> it will load the details of second event.
28. Now lets make each event thumbnail on event details page clickable so that it can redirect us to event details page automatically. To do that we just need to add [routeLink] to <event-thumbnail> element.
29. So go to file “event-thumbnail.component.html” and add [routerLink] like below:
30. 
31. Now if you go to “/events” page and click on any event thumbnail system will automatically load the corresponding event’s details for you.
32. Now lets wire up the “All Events” link in the “navbar component” so that it takes back to “/events” page from “event details” page. So open the file “nav-bar.component.html” and add [routerLink] like below:
33. 
34. Now if you click on “All Events” link in the nav-bar while on event details page you will be taken back to “/events” page.
35. We will now try and navigate from code. For that we will create a small component that we will expand to create events. So lets go create that component first.
36. Right click on “events” folder and click on “New File”. Name the file “create-event.component.ts”.
37. In this file add basic component shell like below:
38. 
39. Add this component in the declarations array in app.module.ts.
40. Now lets add a route that will render the component like below:
41. 
42. Now lets wire up the “Create Event” link in the navigation bar component. So open the file “nav-bar.component.html” and look for link titled as “Create Event”. Add [routerLink] to this link like below:
43. 
44. Now if you click on “Create Event” link in the nav bar it will take you to this create event component and your application will look like below:
45. 
46. Now lets us try and navigate from code. When user click cancel button on the page shown above we want user to be redirected to event details page.
47. For that we will have to import Angular’s Router service and inject it in our code like below. Go to “create-event.component.ts” file:
48. 
49. Now define a “cancel” method that does the redirection and wire it up with cancel button like below:
50. 
51. With this now if you click the cancel button on create event page it will take you back event details page.
52. Now we will want to protect some routes using something called route guards. For example if user enters an event ID (e.g. 42) which does not exists in the URL <http://localhost:4200/events/42> our page will look like below:
53. 
54. As you can notice this is not looking good. So to prevent this what we want to do is we want to redirect the user to a 404 page. So lets create the 404 page.
55. Right click the app folder and click “New Folder”. Name the folder “errors”.
56. Right click the “errors” folder and click “New File”. Name the file as “404.component.ts”.
57. Lets grab the component code from the “misc” folder of cloned or downloaded Github repo.
58. Paste the code in file “404.component.ts” file.
59. Add the component to “declarations” array of app.module.ts file.
60. Now lets add a route for it in the file app-routing.module.ts like below:
61. 
62. Now we will create the route guard service that will redirect us to this 404 component whenever a particular event is not found.
63. Right click the “events” folder and create a file. Name the file “event-route-activator.service.ts”.
64. Add basic class shell into this file like below:
65. 
66. Now implement “CanActivate” interface like below:
67. 
68. What we have done in above code is using the “EventService” we have checked if the event exists or not. If not then we have just redirected the user to 404 page.
69. Lets not forget to add this service as a provider in the providers array of app.module.ts file. So go to app.module.ts and add the service in the providers array like below:
70. 
71. Lets us now wire this service up with the “canActiavte” route guard in our route in app.routing.ts file. So open the app.routing.ts file and attach the route guard like below:
72. 
73. If you now try to hit the URL <http://localhost:4200/events/42> you will see the following page:
74. 
75. Just like we used “canActivate” to prevent user from going to a page we can prevent user from leaving a page by using “canDeactivate” route guard. To use it lets attach “canDeactivate” to our route like below. We want to prevent user from leaving the “New Event” page if they have not saved the event.
76. 
77. Now lets add this function to provider array in app.module.ts. So go to app.module.ts file and in provider array write code like below:
78. 
79. Now lets the define the function in the app.module.ts file like below:
80. 
81. Now since we are checking isDirty property of the “CreateEventComponent” lets define the property in the component. Go to file “create-event.component.ts” and define the property like below:
82. 
83. Now if you refresh the application and go to “Create New Event” page and click “Cancel” button a confirm dialog will pop-up. If you click “Ok” on this dialog system will redirect you to events list page and if click cancel system will keep you on this page only.
84. 