

Sarath Chandra kunisetty – skkh2@umsystem.edu

GitHub link: <https://github.com/kunisettySarath/WebMobileProgramming-Spring22/tree/main/WebDevelopment/ICP's/ICP5>

Venu Linga – vl6hw@umsystem.edu

GitHub link: https://github.com/VenubabuLinga/WebDevCourse/tree/main/WEB_DEV/ICP5

ICP 5

Objective:

The objective of this ICP5 is to develop the webpage using angular components and NgModules..

ToDo App :

Here we created the webpage which is used to add new tasks and illustrates the to do tasks. We do insert the delete button for the same to delete the task from the list of tsks that we inserted.

Index.html:

Here we are calling the app root component.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ToDoTask</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css" integrity="sha384-UHRT">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

app.component.html :

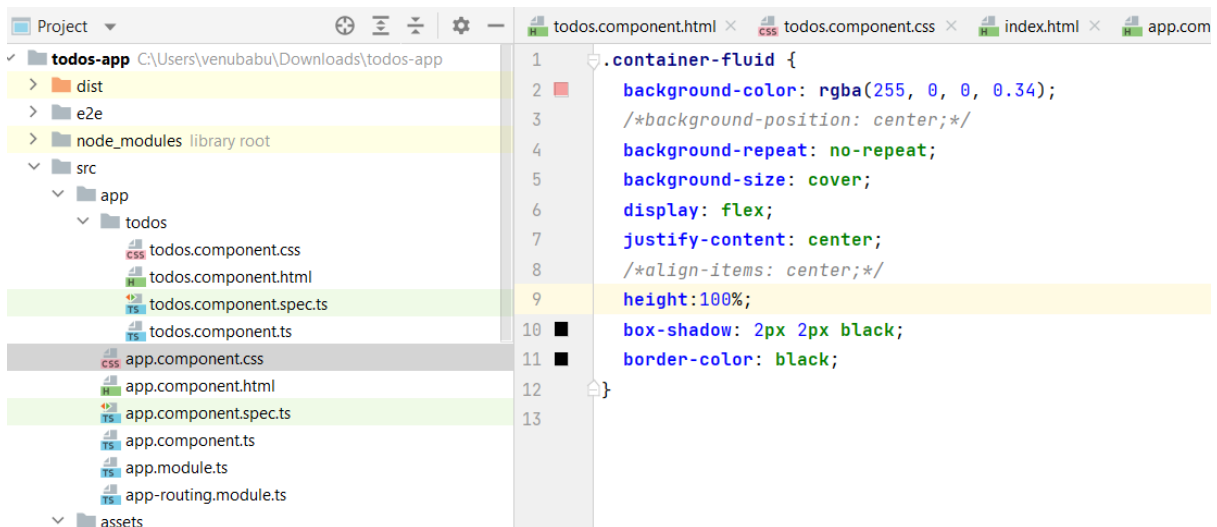
in the app component we added container class where, the **todos** component is called. The **todos** new component was created using command **ng generate component todos**

there fore the new component todos was added with required files like todos.html, todos.css, todos.ts files. As shown in the below snap.



App.component.css :

Here we make the page background colour, border paddings and aligned the content to center.



todos.component.html :

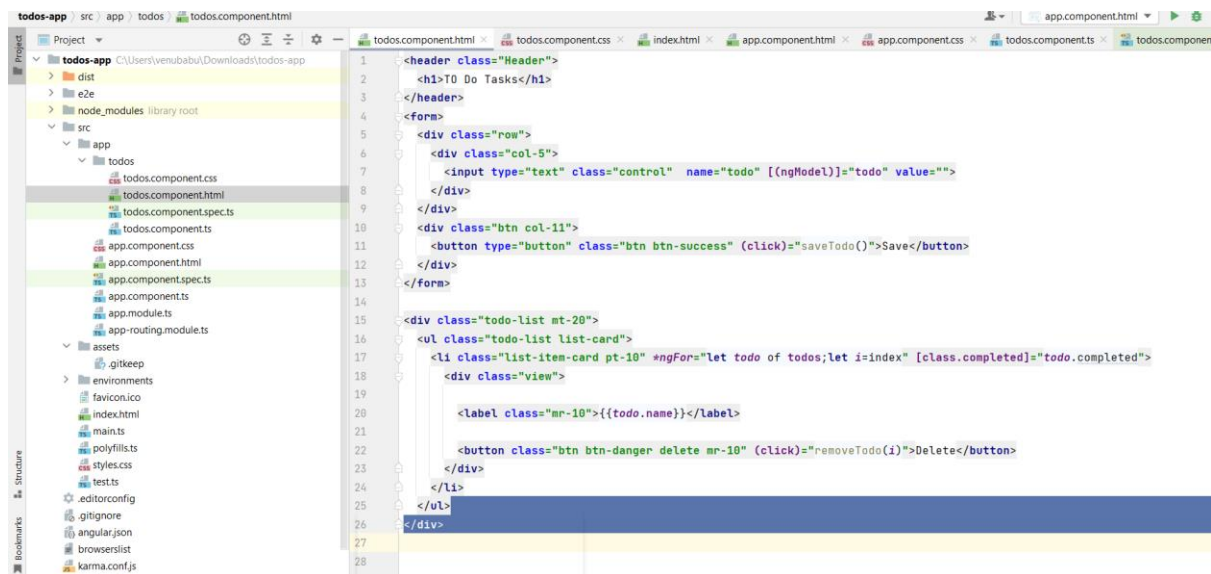
in the todos.component.html page we added the class header class for heading and we do add the form with the input text field, followed by the save button to save the tasks with function savetodo() after the click.

We do added the list to display all the tasks added and also button in it to delete the task.

Todos.component.css:

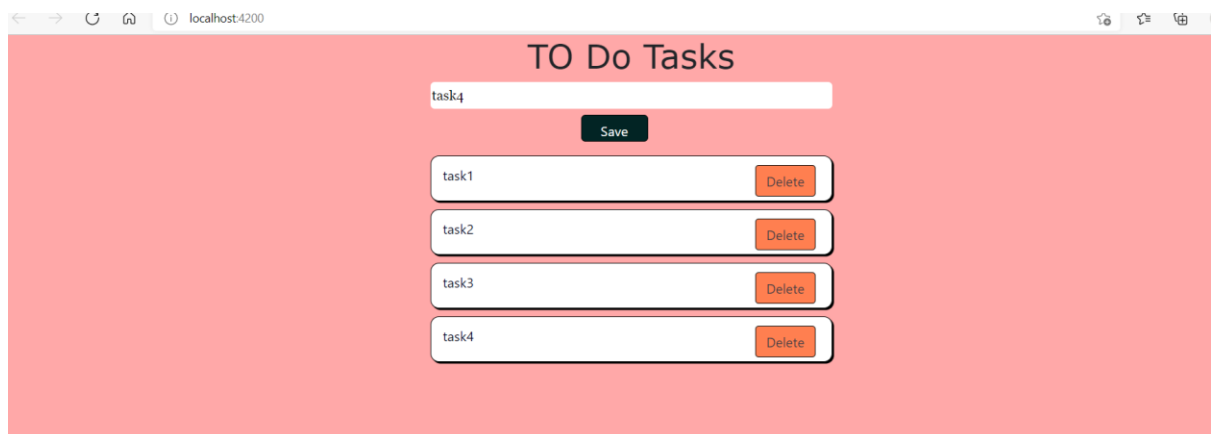
Here we added the required styles for the elements in the todos.component.html.

Input code:



```
1 <header class="Header">
2   <h1>TO Do Tasks</h1>
3 </header>
4 <form>
5   <div class="row">
6     <div class="col-5">
7       <input type="text" class="control" name="todo" [(ngModel)]="todo" value="">
8     </div>
9   </div>
10  <div class="btn col-11">
11    <button type="button" class="btn btn-success" (click)="saveTodo()">Save</button>
12  </div>
13 </form>
14
15 <div class="todo-list mt-20">
16   <ul class="todo-list list-card">
17     <li class="list-item-card pt-10" *ngFor="let todo of todos; let i=index" [class.completed]="todo.completed">
18       <div class="view">
19         <label class="mr-10">{{todo.name}}</label>
20
21         <button class="btn btn-danger delete mr-10" (click)="removeTodo(i)">Delete</button>
22       </div>
23     </li>
24   </ul>
25 </div>
```

Output:



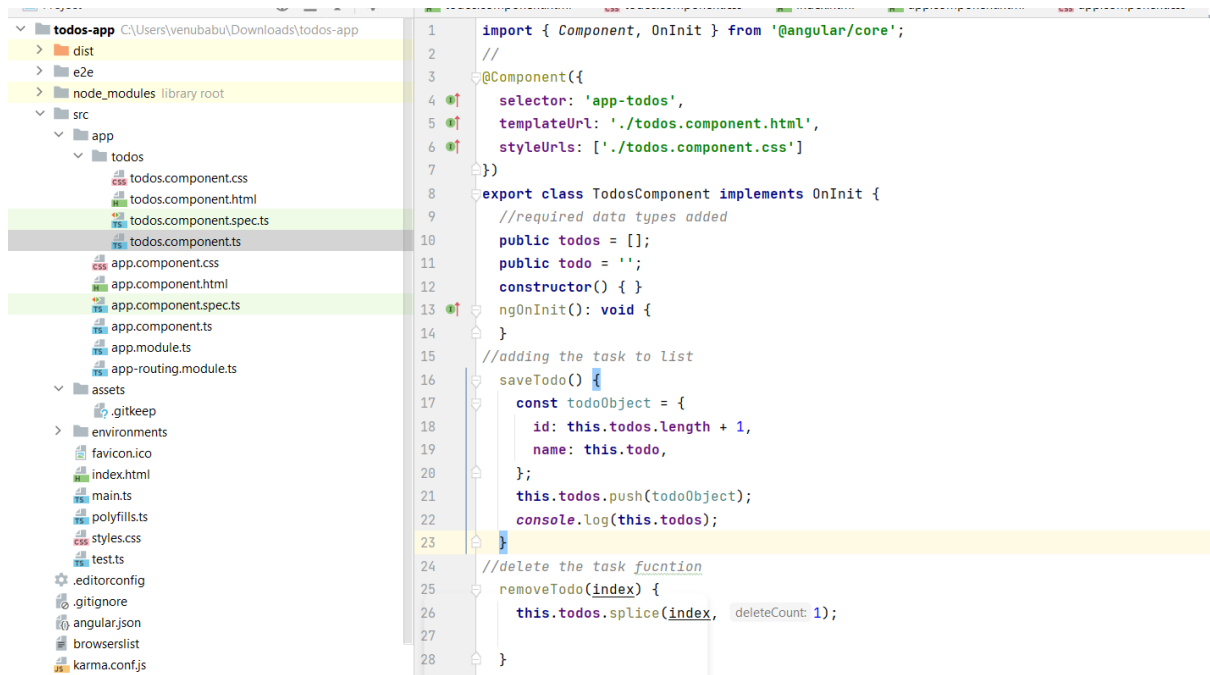
Todos.component.ts:

In this type script file we have added the main scripts to add the new task to list and also functionality to delete the task from the lists.

Adding the task --saveTodo()

Removing the task---removeTodo(index)

Todos.component.ts: input code :

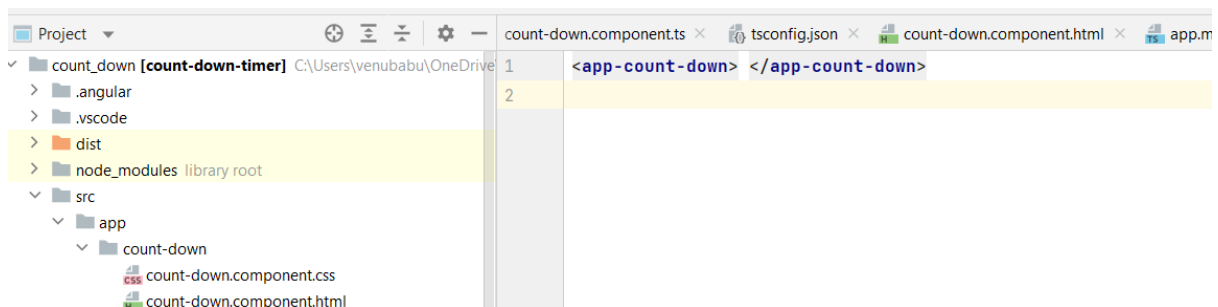


```
1 import { Component, OnInit } from '@angular/core';
2 //
3 @Component({
4   selector: 'app-todos',
5   templateUrl: './todos.component.html',
6   styleUrls: ['./todos.component.css']
7 })
8 export class TodosComponent implements OnInit {
9   //required data types added
10  public todos = [];
11  public todo = '';
12  constructor() {}
13  ngOnInit(): void {
14  }
15  //adding the task to list
16  saveTodo() {
17    const todoObject = {
18      id: this.todos.length + 1,
19      name: this.todo,
20    };
21    this.todos.push(todoObject);
22    console.log(this.todos);
23  }
24  //delete the task function
25  removeTodo(index) {
26    this.todos.splice(index, deleteCount: 1);
27  }
28 }
```

COUNT DOWN App :

Here we have added the new component **count-down**. In this application we are calling the count-down component from index.html and then from the app.component.html.

In the app.component.html we are calling the count-down component.



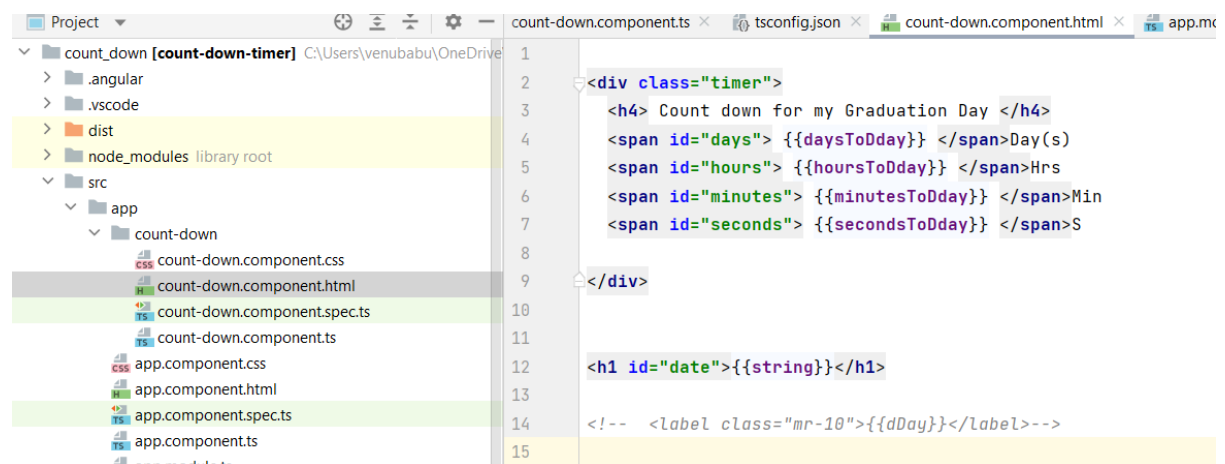
```
1 <app-count-down> </app-count-down>
2
```

Count down.component.html:

In this page we are displaying the count for the graduation day event. With simple output by hard coded dat in type script.

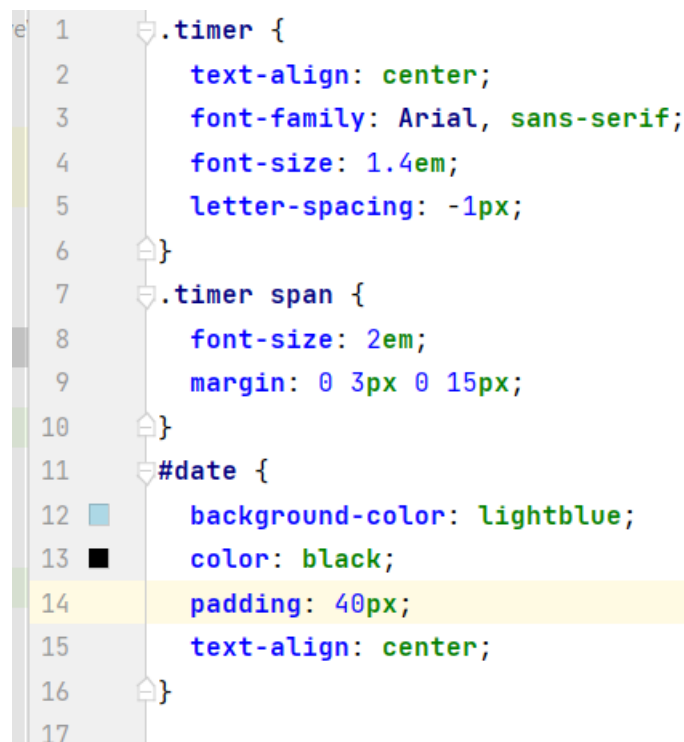
We do added the styling of the page in the count-down.component.css.

Count-down.component.html input code :



```
1
2 <div class="timer">
3   <h4> Count down for my Graduation Day </h4>
4   <span id="days"> {{daysToDday}} </span>Day(s)
5   <span id="hours"> {{hoursToDday}} </span>Hrs
6   <span id="minutes"> {{minutesToDday}} </span>Min
7   <span id="seconds"> {{secondsToDday}} </span>S
8
9 </div>
10
11
12 <h1 id="date">{{string}}</h1>
13
14 <!-- <label class="mr-10">{{dDay}}</label>-->
15
```

Count-down.component.css :



```
1 .timer {
2   text-align: center;
3   font-family: Arial, sans-serif;
4   font-size: 1.4em;
5   letter-spacing: -1px;
6 }
7 .timer span {
8   font-size: 2em;
9   margin: 0 3px 0 15px;
10 }
11 #date {
12   background-color: lightblue;
13   color: black;
14   padding: 40px;
15   text-align: center;
16 }
17
```

Count-down.component.ts :

Here we have hard coded the variables and date as follows and initiated the `dateNow` class.

```
export class CountdownComponent implements OnInit

    private subscription: Subscription;

    public dateNow = new Date();
    //input date for the count down to date
    public dDay = new Date( value: 'Apr 24 2023 00:00'
    milliSecondsInASecond = 1000;
    hoursInADay = 24;
    minutesInAnHour = 60;
    SecondsInAMinute = 60;
```

And also we have casted the date into string and we are printing the string in main page.

We have used the `getTimeDifference()` to get the date difference from today to the hard coded date `dDay`.

We added the `allocateTimeUnits()` to allocate the units of time to display.

```
    public string = this.dDay.toString();
    public timeDifference;
    public secondsToDday;
    public minutesToDday;
    public hoursToDday;
    public daysToDday;
    // taking the date differences funtion
    private getTimeDifference () {
        this.timeDifference = this.dDay.getTime() - new Date().getTime();
        this.allocateTimeUnits(this.timeDifference);
    }

    private allocateTimeUnits (timeDifference) {
        this.secondsToDday = Math.floor( x: (timeDifference) /
            (this.milliSecondsInASecond) % this.SecondsInAMinute);
        this.minutesToDday = Math.floor( x: (timeDifference) /
            (this.milliSecondsInASecond * this.minutesInAnHour) % this.SecondsInAMinute);
        this.hoursToDday = Math.floor( x: (timeDifference) /
            (this.milliSecondsInASecond * this.minutesInAnHour * this.SecondsInAMinute) % this.hoursInADay);
        this.daysToDday = Math.floor( x: (timeDifference) /
            (this.milliSecondsInASecond * this.minutesInAnHour * this.SecondsInAMinute * this.hoursInADay));
    }

    ngOnInit() {
        this.subscription = interval( period: 1000)
            .subscribe( next: x => { this.getTimeDifference(); });
    }
```

OutPut :

