Srikanth Gude – sgcnm@umsystem.edu

Git Hub Link - https://github.com/gudesrikanth/webcourse/tree/main/Webpart/ICP5

Gopi Nelluri – gng75@umsystem.edu

Git Hub Link -
https://github.com/gopinelluri9/demo_remote_repository/tree/main/WebPart/ICP5

ICP – 5(Angular)

Introduction:

Angular is a typescript based full stack and open-source framework for the web applications. Angular is used to create web applications for all types of platforms. It basically agrees the dynamic loading of the web page so, particularly uses is to creation dynamic web applications i.e., Where the material and few components are presented based on the user and the client (web or mobile) that is being consumed. The platform is built to let you create and manage shared code while also dividing tasks among appropriate roles.

Because it reduces the need for extraneous code, Angular provides easy development. It offers a simpler MVC architecture that eliminates the need to write getters and setters. Because directives are not part of the app code, they can be controlled by another team.

Microsoft created and maintained the **TypeScript** programming language. It's a rigorous syntactical superset of JavaScript with the addition of optional static typing. TypeScript is a large-scale application development language that trans compiles to JavaScript.

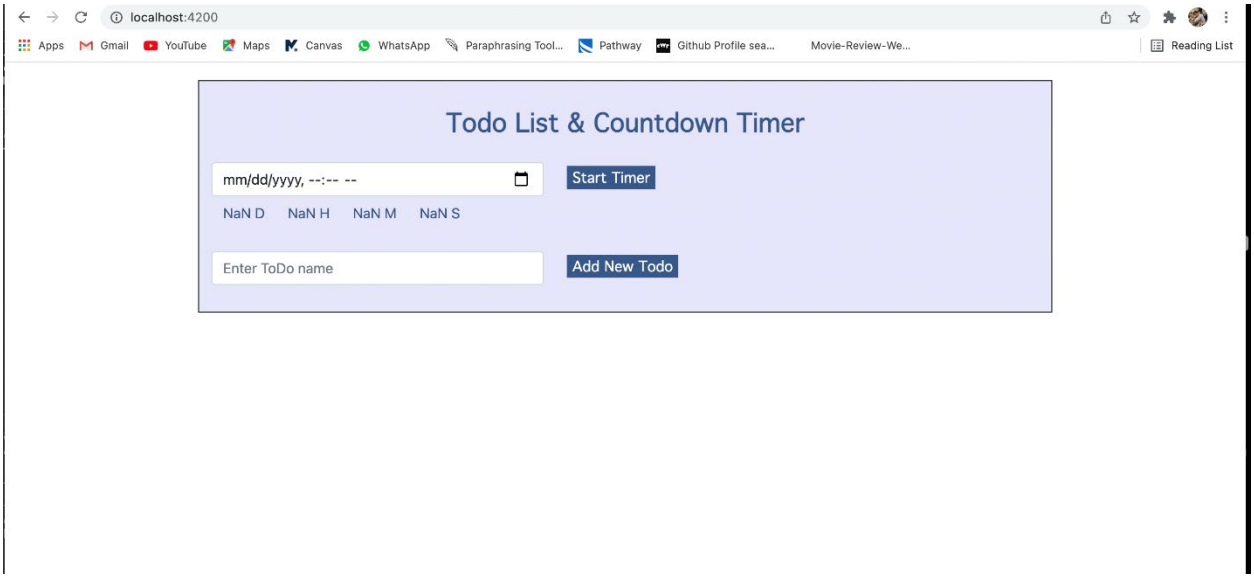
Installation Procedure:

- First, we must install node.js into our system.
- After that, we must install Angular CLI by following next two steps,
- Open the command prompt.
- Type the command "**npm install -g @angular/cli**"
- For creating new Angular project, we must use the command "**ng new yourappname**".
- For running the Angular project, we must use the command "ng serve".

Task1

Webpage Description:

We have created a basic To-Do List application by using Angular and its components (components, string interpolation, property binding, event, and two-way data binding, NgModules, and directives). The web page looks like below,



HTML Code:

```html
webtaskicp5.component.html  webtaskicp5.component.ts  webtaskicp5.component.css  tsconfig.json  icp5.component.css
      </div>
    </div>
  </form>
  <br>
  <!-- Form for the todo user input-->
  <form>
    <div class="row">
      <div class="col-5">
        <input type="text" class="form-control" placeholder="Enter ToDo name" name="todo" [(ngModel)]="todo">
      </div>
      <div class="col-2">
        <button type="button" class="save-button" (click)="saveTodo()">Add New Todo</button>
      </div>
    </div>
  </form>
  <div class="todo-list mt-20">
    <ul class="todo-list list-card">
      <li class="list-item-card pt-10" *ngFor="let todo of todos;let i=index" [class.completed]="todo.completed">
        <div class="view">
          <input class="toggle mr-10" type="checkbox" (click)="toggleTodoComplete(i)" [checked]="todo.completed">
          <b><label class="mr-10 todoName">{{todo.name}}</label></b>
          <button class="btn btn-danger delete mr-10" (click)="removeTodo(i)">Delete</button>
        </div>
      </li>
    </ul>
  </div>
</div>
</body>
</html>
```

ngModel: It is a directive which binds the input and save the user response in a variable. In below code, ngModel is used to store the user value into “todo” variable and adds the given information to list.

TS code:

```typescript
import { Component, OnInit } from '@angular/core';
//Referring the components here
@Component({
  selector: 'app-icp5-web-task',
  templateUrl: './icp5-web-task.component.html',
  styleUrls: ['./icp5-web-task.component.css']
})
```

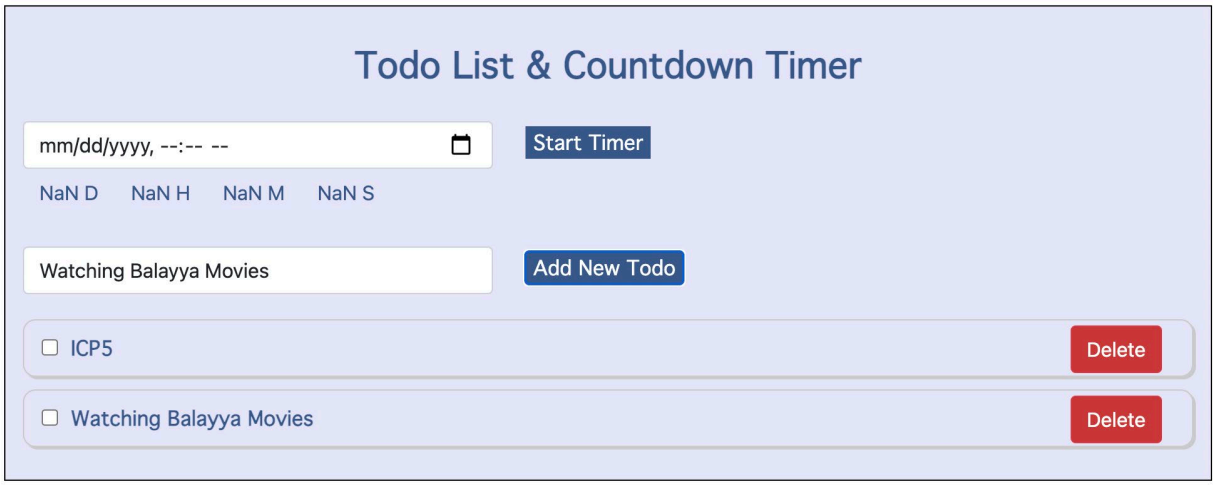
Here @component decorator is helped us to include html and css files.

TS Code:

```
//To Save the ToDo entered by the user
  saveTodo(){
    const todoObject = {
      id: this.todos.length + 1,
      name: this.todo,
      completed: false,
    }
    this.todos.push(todoObject);
    console.log(this.todos)
  }
```

In above code snippet saveTodo() will store to-do list entered by the user and displayed in given order.

Output:



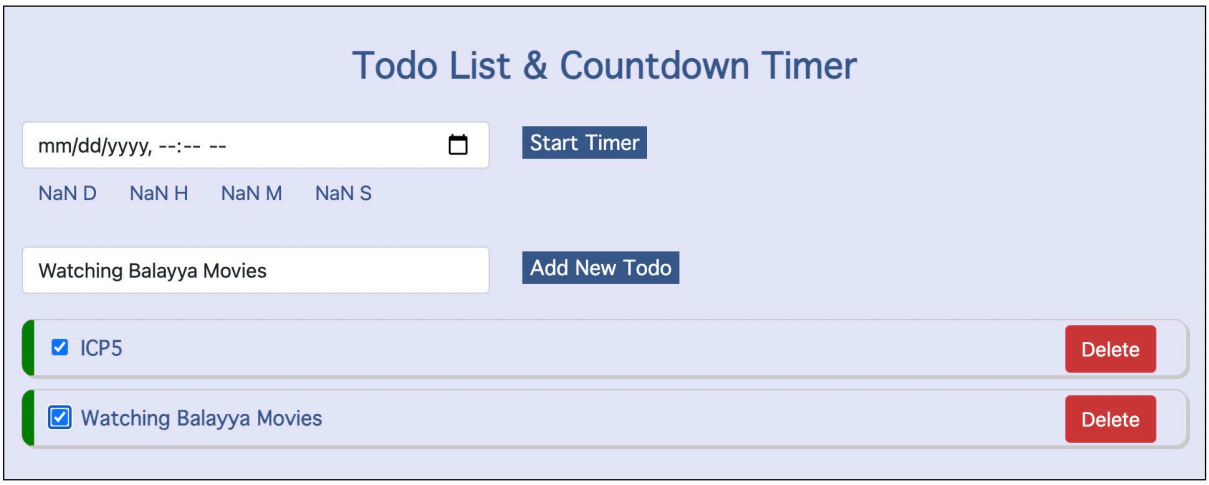
In above page it is clearly showing that, the application saving the to-do lists.

TS Code:

```
  }
//To mark it as completed but not deleted
  toggleTodoComplete(index) {
    const todo = this.todos[index];
    todo['completed'] = !todo['completed'];
    this.todos[index] = todo;
  }
}
```

In above screenshot toggleTodoComplete() is used to mark the completed todo list with checkbox and after marking check box as completed it will display in green color.

Output:



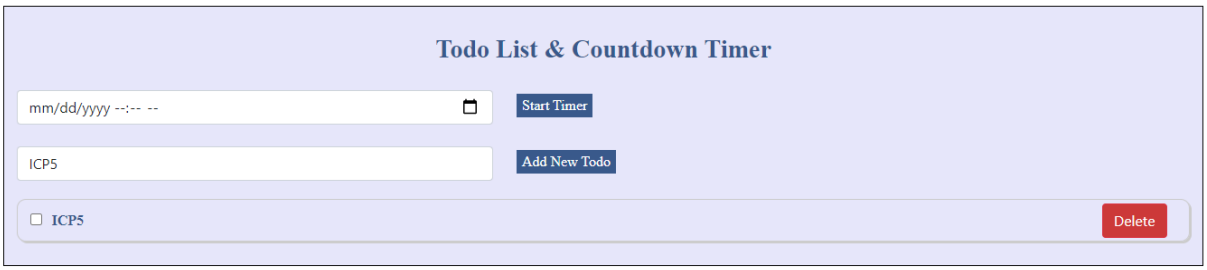
In above page it is clearly showing that, the application marking the to-do list as completed when it has been done.

TS Code:

```
//To remove the ToDo name from the list
  removeTodo(index) {
    this.todos.splice(index, deleteCount: 1)
    this.countDownTimes.splice(index,1)
  }
```

In above code snippet by using removeTodo() method, the web page will delete to-do list stored by the user by clicking delete button.

Output:

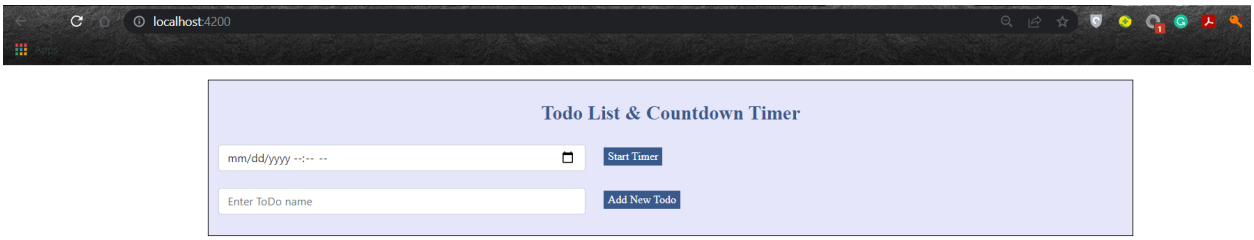


In above page it is clearly showing that, the application removing the to-do lists after clicked on delete button.

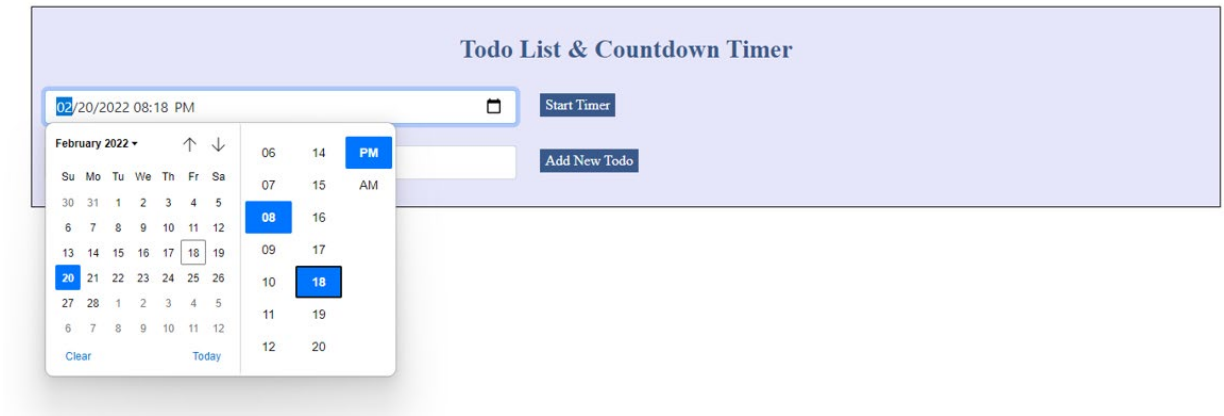
Task – 2

Created a simple countdown timer application utilizing the Angular elements presented and utilized in the Use Case. The Countdown Timer's purpose is to display the months, days, hours, minutes, and seconds until a user-entered event occurs.

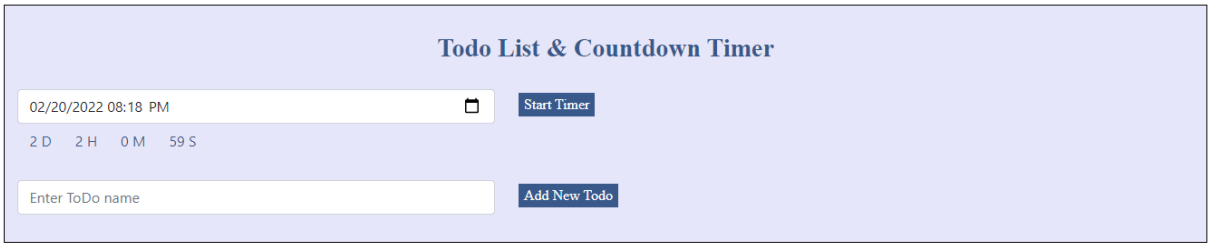
Initial Page:



Select Date for the timer to start:



Click on the strat timer and time is displayed:



HTML Code:

```html
<form>
  <div class="row">
    <div class="col-5">
      <input type="datetime-local" class="form-control" id="tasktime" [(ngModel)]="time" name="time">
    </div>

    <div class="col-2">
      <button type="button" class="save-button" (click)="startTimer()">Start Timer</button>
    </div>
    <br>
    <!-- Displaying Time after starting Timer -->
    <div *ngIf="countDownTimes">
      <button class="btn time">{{countDownTimes.days + ' D'}}</button>
      <button class="btn time">{{countDownTimes.hours + ' H'}}</button>
      <button class="btn time">{{countDownTimes.minutes + ' M'}}</button>
      <button class="btn time">{{countDownTimes.seconds + ' S'}}</button>
    </div>
  </div>
</form>
<br>
```

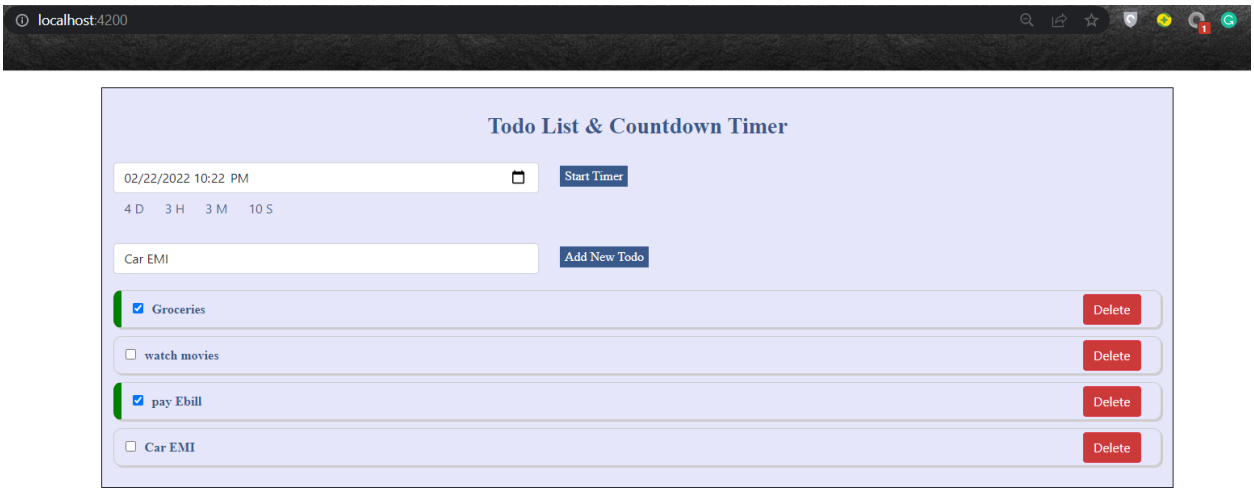
In above screenshot we have used multiple html tags and css styles and some typescript functions are used. **ngModel** is used to bind the time input given by the user and displayed in specific format. **Ngif** is used to count the time in all if and else conditions. And **countDownTimes** days, hours, minutes, seconds buttons is displayed using string interpolation method.

TS code:

```
getCountDown(begin_time) {
  this.interval = setInterval( handler: () => {
    const currentTime = new Date().getTime();
    const difference = begin_time.getTime() - currentTime;
    const days = Math.floor( x: difference / (1000 * 60 * 60 * 24));
    const hours = Math.floor( x: (difference % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
    const minutes = Math.floor( x: (difference % (1000 * 60 * 60)) / (1000 * 60));
    const seconds = Math.floor( x: (difference % (1000 * 60)) / 1000);
    this.countDownTimes= {
      days,
      hours,
      minutes,
      seconds
    };
  }, timeout: 1000);
}
//startTimer function to save the countdown timer
startTimer(){
  clearInterval(this.interval);
  this.getCountDown(new Date(this.time));
}
```

In above code snippet startTimer() function is used to call and get information and perform the action by user. getCountDown method is used to select the date and will display the timer according to user input, in this method we used **this** concept to use functionalities of interval. And Initialized some variables to store the timer input entered by user, and Math.floor() will return the integer values. countDownTimes variable will store and display in given format.

Output:



Conclusion:

In this ICP we learned about how to develop a web page by using the Angular, TypeScript, nodejs and at last, we have designed and developed web page successfully. We didn't face any major issues while doing this ICP.