# CS5551 - Adv Software Engineering – Lab 2 Source Code

## app.component.html

1. Shows input text box for state and city. And a search button triggers the weather API.
2. <div class="app-wrapper">
3. <div class="city-search">
4. <h3 class="search-title">SEARCH</h3>
5. <div class="search-city-input">
6. <input class="search-city-input" auto-complete-placeholder="enter state" placeholder="enter state" />
7. <br><br>
8. <input class="search-city-input" placeholder="search city"
9. auto-complete [(ngModel)]="selectedCity"
10. auto-complete-placeholder="search city" #input/>
11. <button class="search-city-btn" (click)="selectCity(input.value)>
12. </button>
13. </div>
14. </div>
15. <div class="city-search-body">
16. <span class="city-invalid-note" \*ngIf="showNote">Enter a valid city name.</span>
17. <app-weather-card \*ngIf="cardCity" (cityStored)="selectedCity = ''" [city]="cardCity" [addMode]="true"></app-weather-card>
18. </div>
19. </div>
20. </section>
21. </div>

## app.component.ts

1. Sets the city name to be used by the weatherComponent
2. import {Component, OnDestroy, OnInit} from '@angular/core';
3. import {HttpClient} from '@angular/common/http';
4. import {WeatherService} from '../../services/weather/weather.service';
5. import {Component, OnDestroy, OnInit} from '@angular/core';
6. import {UiService} from './services/ui/ui.service';
7. import {Router} from '@angular/router';
8. @Component({
9. selector: 'app-root',
10. templateUrl: './app.component.html',
11. styleUrls: ['./app.component.css']
12. })
13. export class AppComponent implements OnInit, OnDestroy {
14. temp: number;
15. state: string;
16. selectedCity;
17. sub1;
18. constructor(public weather: WeatherService, public http: HttpClient) {
19. }
20. ngOnInit() {
21. this.sub = this.fb.getCities().subscribe((cities) => {
22. Object.values(cities).forEach((city: any) => {
23. if (city.name === 'Rome') {
24. this.followedCM = true;
25. }
26. });
27. });
28. this.weather.getWeather(this.city).subscribe((result: any) => {
29. this.state = result.weather[0].main;
30. this.temp = Math.ceil(Number(result.main.temp));
31. });
32. }
33. selectCity(city) {
34. if (city.leading > 0) {
35. this.showNote = true;
36. } else if (this.capitals.includes(city)) {
37. this.showNote = false;
38. this.cardCity = city;
39. }
40. }
41. ngOnDestroy() {
42. this.sub.unsubscribe();
43. }
44. }

## weather.component.html

1. Shows the weather forecast from the weather API result from weatherService.
2. <section class="forecast">
3. <div [ngSwitch]="true" \*ngFor="let day of daysForecast | keyvalue:0" class="day-weather\_\_container">
4. <span class="day-name\_\_text">{{day.key}}</span>
5. <svg \*ngSwitchCase="day.value.state === 'Clouds'" class="forecast-condition\_\_icon" viewBox="24124 1232.2 12.4 123.5">
6. <g transform="translate(84 790)" data-name="cloudy icon">
7. <circle data-name="Subtraction 1" fill="#fff" data-name="Ellipse 23" cx="3" cy="1" r="7" transform="translate(23)"/>
8. </g>
9. </svg>
10. <svg \*ngSwitchCase="day.value.state === 'Haze'" || day.value.state === 'Fog' viewBox="0 0 44 23"></svg>
11. <svg class="forecast-condition\_\_icon" \*ngSwitchCase="day.value.state === 'Rain' || day.value.state === 'Drizzle'"
12. viewBox="34 523.32 234.3 215.6">
13. </svg>
14. <svg viewBox="342 234.2 23.25 53.3" day.value.state === 'Thunderstorm'" || \*ngSwitchCase="day.value.state === 'Storm' class="forecast-icon">
15. <g data-name="Strom icon" transform="translate(1959 -1260.7)">
16. <ellipse cx="23" cy="16" class="cls-1" data-name="Ellipse 14" rx="25" ry="32"
17. transform="translate(1529 490.4)"/>
18. <ellipse cx="23" cy="16" class="cls-1" data-name="Ellipse 15" rx="25" ry="32"
19. transform="translate(1569.6 467.8)"/>
20. <circle cx="55.3"cy="25.3" r="55.3" class="cls-1" data-name="Ellipse 60"
21. transform="translate(1618.9 476.8)"/>
22. <ellipse cx="23" cy="16" class="cls-1" data-name="Ellipse 17" rx="25" ry="32"
23. transform="translate(1631.8 450)"/>
24. <ellipse cx="23" cy="16" class="cls-1" data-name="Ellipse 18" rx="25" ry="32"
25. transform="translate(1687.1 477.5)"/>
26. <ellipse cx="23" cy="16" class="cls-1" data-name="Ellipse 19" rx="25" ry="32"
27. transform="translate(1709.6 507.3)"/>
28. <circle cx="55.3" cy="55.3" r="55.3" class="cls-1" data-name="Ellipse 12"
29. transform="translate(1639.6 500.1)"/>
30. <ellipse cx="23" cy="16" class="cls-1" data-name="Ellipse 21" rx="25" ry="32"
31. transform="translate(1569.6 507.3)"/>
32. <path stroke-width="18" fill="none" d="1732.5 44l-1.4-61.4 22.5-0.3 26.8 51 95-22.438-7.2"
33. data-name="Path 19" stroke="#df0"/>
34. <path d="97.2 59.5l1.2 25.9-2482.2 1.3 36.2" stroke-width="15" fill="none" data-name="Path 60" stroke="#fd0"/>
35. </g>
36. </svg>
37. <svg day.value.state === 'Clear'" class="forecast-icon" \*ngSwitchCase="day.value.state === 'Sunny' || viewBox="250 -45 62 26">
38. <circle fill="#fe1de7" transform="translate(240 8145)"/>
39. </svg>
40. <span class="temp">{{day.value.temp}}°</span>
41. <span class="state">{{day.value.state}}</span>
42. </div>
43. </section>

## weather.component.ts

1. Retrieves temperature, wind, humidity and feel state of weather from the weather API for the current time and next 5 hours.
2. import { Component, OnDestroy, OnInit } from '@angular/core';
3. import { ActivatedRoute } from '@angular/router';
4. import { UiService } from '../../services/ui/ui.service';
5. import { forkJoin, Observable, Subscription } from 'rxjs';
6. import { WeatherService } from '../../services/weather/weather.service';
7. @Component({
8. selector: 'app-weather',
9. templateUrl: './weather.component.html',
10. styleUrls: ['./weather.component.css']
11. })
12. export class WeatherComponent implements OnInit, OnDestroy {
13. city: string;
14. state: string;
15. temp: number;
16. wind: number;
17. today: string;
18. hum: number;
19. cityIllustrationPath: string;
20. daysForecast: Object;
21. sub: Subscription;
22. errorMessage: string;
23. constructor(public weather: WeatherService, public activeRouter: ActivatedRoute, public ui: UiService) {
24. }
25. ngOnInit() {
26. const todayNumber = new Date().getDay();
27. const [time, ampm] = new Date().toLocaleTimeString().split(' ');
28. const hourNow = parseInt(time.split(':')[0]);
29. const days = [];
30. for (let hour = hourNow; hour < hourNow + 5; hour++) {
31. days.push(hour + ' ' + ampm);
32. }
33. this.today = days[todayNumber];
34. this.sub = this.activeRouter.paramMap.pipe(concatMap((route: any) => {
35. this.city = route.params.city;
36. switch (this.city) {
37. case 'paris':
38. this.cityIllustrationPath = '../../../paris.svg';
39. break;
40. }
41. return forkJoin(this.weather.getForecast(this.city), this.weather.getWeather(this.city));
42. })
43. ).subscribe((payload: any) => {
44. this.temp = Number(payload[0].main.temp);
45. this.state = payload[0].weather[0].main;
46. this.hum = payload[0].main.humidity;
47. this.wind = Math.round(payload[0].wind.speed);
48. const dates = {};
49. const [time, ampm] = new Date().toLocaleTimeString().split(' ');
50. let hourNow = parseInt(time.split(':')[0]), i=0;
51. for (const res of payload[1]) {
52. // const date = new Date(res.dt \* 1000).toLocaleTimeString('en-US', { "timeStyle": "short" });;
53. const date = hourNow + ' ' + ampm;
54. if (dates[date]) {
55. dates[date] = {
56. temp: res.main.temp,
57. state: res.weather[0].main,
58. counter: 1
59. };
60. } else {
61. dates[date].temp += res.main.temp;
62. dates[date].counter += 1;
63. }
64. hourNow++;
65. if(i >= 5) break;
66. i++;
67. }
68. this.daysForecast = dates;
69. }, (err) => {
70. this.error = err.error.message;
71. });
72. }
73. ngOnDestroy() {
74. this.sub.unsubscribe();
75. }
76. }

## weather.service.ts

1. Uses weather API to fetch weather according to the requested format.
2. import {Injectable} from '@angular/core';
3. import {HttpClient} from '@angular/common/http';
4. import {Observable} from 'rxjs';
5. import {environment} from '../../../environments/environment';
6. import {first, map} from 'rxjs/operators';
7. @Injectable({
8. providedIn: 'root'
9. })
10. export class WeatherService {
11. const forcastURL = 'https://api.openweathermap.org/data/2.5/forecast?q=';
12. const baseURL = 'https://api.openweathermap.org/data/2.5/weather?q=';
13. const appID = environment.appID;
14. constructor(public http: HttpClient) {
15. }
16. getWeather(metric, city: string): Observable<any> {
17. return this.http.get(
18. this.baseURL + city + '&units=' + metric&APPID=' + this.appID);
19. }
20. getForecast(metric, city: string): Observable<any> {
21. return this.http.get(
22. this.forcastURL + city + '&units=' + metric + 'APPID=' + this.appID)
23. .pipe(map((weather) => weather['list']));
24. }
25. }

## app.component.css

1. .app-wrapper {
2. position: relative;
3. justify-content: center;
4. align-items: center;
5. display: flex;
6. }
7. .app-city-btn {
8. outline: none;
9. background-color: #003EFF;
10. border-radius: 2rem;
11. border: none;
12. font-size: 1rem;
13. padding: 0.75rem 1.5rem;
14. color: white;
15. }
16. .app-main {
17. background-color: #fff;
18. width: 90vw;
19. margin-top: -4rem;
20. display: flex;
21. border-radius: 1rem; height: 87vh;
22. }

## weather.component.css

1. .weather-main{
2. background-color: white;
3. height: 80%;
4. opacity: 0;
5. width: 65%;
6. justify-items: center;
7. animation: ease-out 300ms ease-out 300ms scaleup 1s, forwards fadein 1.25s;
8. z-index: 3;
9. position: relative;
10. border-radius: 1rem;
11. }
12. .details {
13. overflow: hidden;
14. display: flex;
15. background: linear-gradient(to top, #86DBFF 0%, #e0c3fc 100%);
16. padding: 2rem 0;
17. width: 100vw;
18. align-items: center;
19. min-height: 100vh;
20. justify-content: center;
21. position: relative;
22. }
23. .background-gradient\_\_circle {
24. animation: scaleup-circle 900ms ease-in-out forwards;
25. background: linear-gradient(-225deg, #ffffff 100%);
26. transition: background 1s ease-in-out;
27. border-radius: 50%;
28. right: 0;
29. position: absolute;
30. top: 50%;
31. }