

**FULLSTACK DEVELOPMENT-1 (21IS551)**

**Activity Based Assessment**

**On**

**“Weather App: City-based Weather Information”**

**Submitted By:**

**Pradeep Shrikanth Ninganuri [4VV21IS073]**

**R Yoganarasimhan [4VV21IS081]**

**Sharath Kumar K R [4VV21IS095]**

**Submitted To:**

**Prof. Kasturi Rangan R**

**Department of**

**Information Science & Engineering**

**VVCE**

**CONTENT**

**--------------**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Title** | **Pg.no** |
| **1** | **Introduction** | **3** |
| **2** | **Angular** | **4** |
| **3** | **Weather app** | **9** |
| **4** | **Weather app using angular** | **10** |
| **5** | **Code and output** | **11** |
| **6** | **Conclusion** | **14** |

**Abstract**

In today's fast-paced world, staying updated with the weather forecast is crucial for planning daily activities, travel arrangements, and ensuring overall safety. This abstract introduces an Angular-based Weather Application designed to provide users with real-time weather updates in a user-friendly interface.

The Angular Weather App utilizes Angular framework's powerful features to create a responsive and interactive web application. Through integration with reliable weather APIs such as OpenWeatherMap, the app fetches accurate and up-to-date weather data for locations worldwide.

**Introduction**

In an era where instant access to accurate weather information is indispensable for daily life, the development of efficient and user-friendly weather applications becomes paramount. The Angular Weather App emerges as a solution to this necessity, leveraging the robust capabilities of Angular framework to deliver a seamless and intuitive platform for obtaining real-time weather updates.

Angular, a popular open-source web application framework maintained by Google, offers a comprehensive toolkit for building dynamic and responsive single-page applications (SPAs). With its component-based architecture and powerful data binding features, Angular provides developers with the tools necessary to create interactive and feature-rich applications.

The Angular Weather App capitalizes on Angular's strengths to deliver a compelling user experience, offering a modern and visually appealing interface coupled with reliable weather data sourced from APIs like OpenWeatherMap. By harnessing Angular's capabilities for asynchronous data retrieval and real-time updates, the app ensures that users have access to the latest weather information for their desired locations.

This introduction sets the stage for exploring the Angular Weather App, highlighting its key features, including its intuitive user interface, real-time data updates, customization options, and responsive design. Through the utilization of Angular's architecture and best practices, the Angular Weather App aims to provide users with a seamless and efficient tool for staying informed about weather conditions, empowering them to plan their activities with confidence and precision.

**Angular**

Angular is a popular open-source front-end web application framework maintained by Google and a community of developers. It is used for building dynamic, single-page web applications (SPAs) and is based on TypeScript, a superset of JavaScript.

Key features of Angular include:

1. Component-based architecture: Angular applications are built using reusable and encapsulated components, which help in organizing the application's structure and logic.

2. Two-way data binding: Angular provides two-way data binding between the model and the view, which allows changes in the model to be reflected in the view automatically, and vice versa.

3. Directives: Angular provides built-in directives like ngIf, ngFor, ngSwitch, etc., which extend HTML with additional functionality and allow developers to create dynamic views.

4. Dependency injection: Angular's dependency injection system helps in managing dependencies between different components and services, making the application more modular and easier to test.

5. Services: Angular services are singleton objects that can be injected into components and other services, providing a way to share data and functionality across the application.

6. Routing: Angular's router allows developers to build single-page applications with multiple views, each mapped to a different URL, enabling navigation within the application without page reloads.

7. Forms: Angular provides powerful support for building forms, including template-driven forms and reactive forms, with features like validation, error handling, and form submission.

8. HTTP client: Angular's built-in HTTP client module simplifies making HTTP requests to backend servers, handling request and response transformations, error handling, and more.

Overall, Angular provides a robust framework for building modern web applications with a strong focus on developer productivity, maintainability, and performance.

**Weather App: City-based Weather Information**

**Introduction**

The Weather App, developed using Angular, is a dynamic web application designed to provide real-time weather information for a specified city. Leveraging the OpenWeatherMap API, the app fetches accurate and up-to-date weather data, featuring temperature, humidity, and wind information—all dynamically updated based on the user's entered city.

**User Interface**

The main display section of the Weather App presents crucial weather details for the chosen city. Users encounter the current date, location, and an iconic representation of the weather. The temperature, prominently displayed in Celsius for clarity, is complemented by a brief weather description, offering a quick overview of the current conditions.

**User Interaction**

The app prioritizes user flexibility, allowing them to dynamically change the location. By clicking the "Change location" button, the input field toggles, facilitating the easy entry of a different city.

**Weather app using Angular :**

Creating a weather app using Angular involves several steps, including setting up the Angular project, fetching weather data from a weather API, and displaying the data in the application. Below is a basic outline of how you can create a simple weather app using Angular:

1. Set up Angular project:

You can use Angular CLI to create a new Angular project

ng new weather-app

cd weather-app

2. Install dependencies:

You may need to install additional dependencies like @angular/http for making HTTP requests or any other libraries you plan to use.

3. Create Service:

Angular services provide a way for you to separate Angular app data and functions that can be used by multiple components in your app.

ng g s servicename

4. Fetch weather data:

Choose a weather API to fetch weather data. Popular choices include OpenWeatherMap, Weatherstack, or Dark Sky. You'll need an API key to access the data.

Use Angular's HTTP client to make requests to the weather API and fetch the weather data.

5. Display weather data:

Once you receive the weather data from the API, you can display it in your Angular components. You might display information such as temperature, humidity, wind speed, etc.

Use data binding in your Angular templates to display the weather data dynamically.

6. Handle user input:

Implement a search feature where users can enter a location to get the weather information for that location.

Use Angular forms for handling user input and triggering the weather data fetch based on the user's input.

7. Styling:

Apply styles to your components and UI to make your weather app visually appealing. You can use CSS frameworks like Bootstrap or Material Design to quickly style your components.

8. Deployment:

Once your weather app is ready, deploy it to a web server or a hosting service so that users can access it online.

This is a basic outline, and there's a lot more you can do to enhance your weather app, such as adding features like geolocation-based weather, hourly forecasts, or integrating with maps for visual representations of weather data.

**Codes:**

**1. App.Component.ts :**

import { Component } from '@angular/core';

import { CommonService } from './common.service';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Weather';

IsChangeLocation=false;

TodayDate: Date;

weatherData: any;

cityName: any;

constructor(private service:CommonService){

this.TodayDate = new Date();

}

changeLocation(){

debugger

this.IsChangeLocation = this.IsChangeLocation==true?false:true;

}

getWeatherDataByCity(){

var city = ("CityName").val();

this.service.getWeatherData(city).subscribe(data=>{

this.cityName = city;

console.log('data',data)

this.weatherData = data;

})

}

transform(value: number): number {

return Math.round(value);

}

}

**2. App.component.html :**

<div class="mainDIv">

<div class="container">

<div class="weather-side">

<div class="weather-gradient"></div>

<div class="date-container">

<h2 class="date-dayname"> {{TodayDate | date:'EEEE'}} </h2> <span class="date-day"> {{TodayDate | date:'MMM y'}} </span> <i class="location-icon" data-feather="map-pin"> </i> <span class="location"> {{cityName}} </span>

</div>

<div class="weather-container"><i class="weather-icon" data-feather="sun"></i>

<h1 class="weather-temp">{{transform(weatherData.main.temp-273.15)}}°C</h1>

<h3 class="weather-desc">{{weatherData.weather[0].description}}</h3>

</div>

</div>

<div class="info-side">

<div class="today-info-container">

<div class="today-info">

<div class="humidity"> <span class="title">HUMIDITY</span><span class="value">{{weatherData.main.humidity}} %</span>

<div class="clear"></div>

</div>

<div class="wind"> <span class="title">WIND</span><span class="value">{{weatherData.wind.speed}} km/h</span>

<div class="clear"></div>

</div>

</div>

</div>

<div class="location-container">

<button class="location-button" type="button" (click)="changeLocation()"> <i data-feather="map-pin"></i><span>Change location</span></button>

</div>

<div>

<input type="text" (change)="getWeatherDataByCity()" id="CityName" class="form-control" placeholder="enter city name">

</div>

</div>

</div>

</div>

**3. Common.service.ts :**

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class CommonService {

constructor(private http:HttpClient) { }

getWeatherData(city:string):Observable<any>{

var headers={

'X-RapidAPI-Key': '34e65fe2f0mshb98fd4f81c19c2cp1e6926jsn4dc2cda5a444',

'X-RapidAPI-Host': 'openweather43.p.rapidapi.com'

}

var q = city;

var appid = ['da0f9c8d90bde7e619c3ec47766a42f4'];

var units = 'standard';

return this.http.get('https://openweather43.p.rapidapi.com/weather',{params:{q,appid,units },headers:headers})

}

}

**4. App.module.ts :**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { HttpClientModule } from '@angular/common/http';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Output:**



**Conclusion:**

In conclusion, the development of a weather app using Angular provides a seamless and efficient way for users to access accurate and up-to-date weather information. Through the use of Angular's robust framework, we were able to create a dynamic and responsive user interface that enhances the overall user experience.

By integrating various APIs and libraries, such as the OpenWeatherMap API and Angular Material, we were able to provide users with comprehensive weather forecasts, including current conditions, hourly forecasts, and extended forecasts. Additionally, features such as geolocation allow for personalized weather updates based on the user's location, further improving the app's usability.

The modular structure of Angular facilitated the development process, allowing for easier maintenance and scalability of the app as new features are added or existing ones are updated. Furthermore, Angular's strong community support and extensive documentation provided valuable resources throughout the development lifecycle.

Overall, the weather app developed using Angular offers a user-friendly interface, accurate weather information, and scalability for future enhancements, making it a valuable tool for users seeking reliable weather forecasts.