5. Creating a Weather App

Creating a Weather App using the MEAN stack (MongoDB, Express.js, Angular, Node.js) is a straightforward project. In this case, MongoDB isn't typically used for this type of application since it's a database system, and weather data is typically fetched from external APIs. Instead, you can focus on building the backend using Node.js and Express.js and the frontend using Angular.

Here's a high-level overview and code snippets to guide you:

Project Setup and Structure

Set up a new project folder and structure for your Weather App. Install the required Node.js packages and create a basic Angular application.

```
# Create a new Angular application
ng new weather-app
```

- ## Backend (Node.js & Express.js)

Create the backend of your Weather App using Node.js and Express.js.

Installation of Packages

Install the necessary packages for Express.js.

npm install express axios

Setting up Express.js

Create your Express.js server and set up middleware.

- **javascript**

```
// server.js

const express = require('express');
const axios = require('axios');

const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());

// Define API route for fetching weather data
app.get('/api/weather', async (req, res) => {
  try {
    const { location } = req.query; // Get location from query params
```

```
    const apiKey = 'YOUR_API_KEY'; // Replace with your weather API key

    // Fetch weather data from a third-party API (e.g., OpenWeatherMap)
    const response = await axios.get(
      `https://api.openweathermap.org/data/2.5/weather?q=${location}&appid=${apiKey}`
    );

    const weatherData = response.data;
    res.json(weatherData);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Could not fetch weather data' });
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

- **Frontend (Angular)**

Create the frontend of your Weather App using Angular. Design the user interface and implement weather data retrieval.

Design and UI

Design the user interface for your Weather App using Angular components, templates, and styles.

Fetch Weather Data

Create a component that allows users to input a location and fetch weather data from the backend.

- **typescript**

```
// weather.component.ts
import { Component } from '@angular/core';
import { WeatherService } from './weather.service';

@Component({
  selector: 'app-weather',
  templateUrl: './weather.component.html',
})
```

```
export class WeatherComponent {
  location: string;
  weatherData: any;

  constructor(private weatherService: WeatherService) {}

  getWeather() {
    this.weatherService.getWeather(this.location).subscribe((data) =>
{
      this.weatherData = data;
    });
  }
}
```

External API

To fetch weather data, you can use a third-party weather API such as OpenWeatherMap, WeatherAPI, or any other provider of your choice. Sign up for an API key and replace 'YOUR_API_KEY' in the server code with your actual API key.

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to the Node.js server. Ensure that you handle

error cases gracefully and format the weather data appropriately for display in your Angular components.

Remember to handle API rate limits, error handling, and user interface design to make your Weather App user-friendly and informative.