26. Weather Dashboard

Creating a Weather Dashboard using the MEAN stack (MongoDB, Express.js, Angular, Node.js) is a useful project to provide users with weather information for multiple locations. Below is a high-level overview of how to build such a dashboard, along with some code snippets to guide you:

Project Setup and Structure

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

```
# Create a new Angular application

ng new weather-dashboard-app
```

- Backend (Node.js & Express.js)

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

Installation of Packages

Install the necessary packages for Express.js, Mongoose (for MongoDB), and other dependencies.

```
npm install express mongoose cors
```

Setting up Express.js

Create your Express.js server, set up middleware, and handle routes.

- **javascript**

```
// server.js

const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();

// Middleware
app.use(express.json());
app.use(cors());

// Database connection (optional if you want to store user preferences)
mongoose.connect('mongodb://localhost/weather-dashboard-app', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
```

```
  useCreateIndex: true,
});

// Define Mongoose models for User and Location data (optional)
const User = mongoose.model('User', {
  username: String,
  password: String, // Use hashing for security
  // Add more user-related fields as needed
});

const Location = mongoose.model('Location', {
  userId: mongoose.Schema.Types.ObjectId,
  name: String,
  // Add more location-related fields as needed
});

// Routes for managing users and locations (optional)
app.post('/api/register', async (req, res) => {
  // Register a new user
  // Store hashed password in the database
});
```

```
app.post('/api/login', async (req, res) => {
  // Authenticate user and generate a JWT token
});

app.post('/api/locations', async (req, res) => {
  // Create a new location entry (optional)
  // Save the location to the database (optional)
});

app.get('/api/locations', async (req, res) => {
  // Retrieve a list of saved locations (optional)
});

// Create a route for fetching weather data from a weather API
app.get('/api/weather', async (req, res) => {
  // Use an external weather API (e.g., OpenWeatherMap, Weather API) to fetch weather data
  // Provide the data to the frontend
});
```

- ## Frontend (Angular)

Create the frontend of your Weather Dashboard app using Angular. Design the user interface for displaying weather information for multiple locations.

Design and UI

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

Weather Display

Create components for displaying weather information, including temperature, conditions, forecasts, and icons.

Location Management (optional)

Design components for users to add, remove, and manage saved locations.

User Authentication (optional)

Implement user registration and login functionality if you choose to allow users to save their favorite locations.

- **typescript**

// weather-display.component.ts

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

@Component({
  selector: 'app-weather-display',
  templateUrl: './weather-display.component.html',
})
export class WeatherDisplayComponent implements OnInit {
  weatherData: any;

  constructor(private weatherService: WeatherService) {}

  ngOnInit() {
    // Fetch weather data from the backend
    this.weatherService.getWeatherData().subscribe((data) => {
      this.weatherData = data;
    });
  }
}
```

MongoDB (optional)

Create a MongoDB database to store user profiles and saved locations if you choose to implement user authentication and location management.

Weather API Integration

Integrate an external weather API (e.g., OpenWeatherMap, Weather API) to fetch weather data for the specified locations. Ensure that you have an API key for access.

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to Node.js routes. Ensure that you display weather data for multiple locations and handle user preferences (optional) properly.

Building a Weather Dashboard is a practical project that provides valuable information to users. You can expand it with features like weather maps, extended forecasts, severe weather alerts, and user customization for a more comprehensive weather tracking experience.