# Weather App – Fullstack Project Plan (Angular 17 + .NET 8 + SQL Server 2022)

## 1. Frontend (Angular 17) – Features

A. Public Features (No Login Required):  
1. Search weather by city name (with search history stored in LocalStorage).  
2. View current weather: °C/°F, condition, humidity, wind speed, weather icon.  
3. View 5-day forecast: date, min/max temperature, weather condition.  
4. Show current location when page loads (HTML5 Geolocation API).  
  
B. Logged-in User Features:  
5. Login / Register / Logout (JWT).  
6. View and update profile information (email, password).  
7. Save favorite cities.  
  
C. Administrator Features:  
8. Add / edit / delete weather data (we do not use OpenWeatherMap API).  
9. Manage city list.  
10. Manage user accounts.

## 2. Backend (.NET 8) – REST API Endpoints

### Auth

|  |  |
| --- | --- |
| Endpoint | Description |
| POST /api/auth/register | Register new account |
| POST /api/auth/login | Login and return JWT |
| POST /api/auth/logout | Logout (optional if JWT stored client-side) |
| GET /api/auth/profile | Get current user profile |

### Cities

|  |  |
| --- | --- |
| Endpoint | Description |
| GET /api/cities | Get all cities |
| GET /api/cities/{id} | Get city details |
| POST /api/cities | Add new city (Admin) |
| PUT /api/cities/{id} | Update city (Admin) |
| DELETE /api/cities/{id} | Delete city (Admin) |

### Weather

|  |  |
| --- | --- |
| Endpoint | Description |
| GET /api/weather/current/{cityName} | Get current weather for a city |
| GET /api/weather/forecast/{cityName} | Get 5-day forecast for a city |
| POST /api/weather | Add weather data (Admin) |
| PUT /api/weather/{id} | Update weather data (Admin) |
| DELETE /api/weather/{id} | Delete weather data (Admin) |

### Favorites

|  |  |
| --- | --- |
| Endpoint | Description |
| GET /api/favorites | Get user's favorite cities |
| POST /api/favorites/{cityId} | Add city to favorites |
| DELETE /api/favorites/{cityId} | Remove city from favorites |

## 3. Database (SQL Server 2022) – Tables

Users:  
- Id (int, PK) – Primary key  
- Username (nvarchar(50)) – Username  
- Email (nvarchar(100)) – Email address  
- PasswordHash (nvarchar(MAX)) – Hashed password  
- Role (nvarchar(20)) – User role (User/Admin)  
- CreatedAt (datetime) – Account creation date  
  
Cities:  
- Id (int, PK) – Primary key  
- Name (nvarchar(100)) – City name  
- CountryCode (nvarchar(10)) – Country code  
- Latitude (float) – Latitude  
- Longitude (float) – Longitude  
  
WeatherForecasts:  
- Id (int, PK) – Primary key  
- CityId (int, FK) – Linked to Cities  
- Date (datetime) – Forecast date  
- Temperature (float) – Average temperature  
- TempMin (float) – Min temperature  
- TempMax (float) – Max temperature  
- WeatherStatus (nvarchar(50)) – Weather condition  
- Humidity (int) – Humidity percentage  
- WindSpeed (float) – Wind speed  
- CreatedAt (datetime) – Record creation date  
- Source (nvarchar(50)) – Data source  
  
Favorites:  
- UserId (int, FK) – Linked to Users  
- CityId (int, FK) – Linked to Cities  
- CreatedAt (datetime) – Date added to favorites

## 4. Development Plan

Phase 1 – Backend Setup:  
1. Create .NET 8 Web API project.  
2. Configure SQL Server connection.  
3. Create DbContext and entity classes (User, City, WeatherForecast, Favorite).  
4. Run initial migration.  
  
Phase 2 – Authentication:  
5. Implement register/login APIs (JWT).  
6. Implement JWT authentication middleware.  
7. Implement role-based authorization.  
  
Phase 3 – Cities:  
8. Implement CRUD APIs for Cities.  
9. Validate input data.  
  
Phase 4 – Weather:  
10. Implement current and forecast weather APIs.  
11. Implement CRUD APIs for Weather data (Admin).  
  
Phase 5 – Favorites:  
12. Implement APIs for managing favorite cities.  
  
Phase 6 – Frontend (Angular 17):  
13. Implement Auth (Login/Logout/Register).  
14. Implement Weather UI (Search, Current, Forecast).  
15. Implement Admin UI (City & Weather management).  
16. Implement Favorites UI.