USER STORY

Weather Application Documentation

Document Version : 1.0

Author : Dharma Rakshak Tadi

Date Updated : 07-19-2024

Table of Contents

1. Overview

This Weather Application provides real-time weather updates for a configured location. Users can configure the location, update interval, and view both recorded historical weather data and historical weather data. The application is built using ASP.NET Core MVC for the overall structure and SignalR for real-time updates.

Users can retrieve the current temperature and weather description by setting the weather configuration, which includes the city name, update interval (in minutes), number of days for recorded historical data, and number of days for historical data. The weather details are updated on the user interface based on the specified update interval.

Users can view the recorded weather details collected by the application and see historical data for a particular city by configuring the values for the city and the number of days for recorded historical data. The recorded historical data is updated each time the weather details are refreshed on the screen, along with the date and time of the update.

Additionally, users can view historical weather data by configuring the location details and the number of days for historical data fields. The historical data for the past configured days is displayed for the selected city.

Key Features:

- **Real-time Weather Updates**: Provides up-to-date weather information without the need for manual refreshes.
- **User Configuration**: Allows users to set the city, update interval, and periods for historical data retrieval.
- **Historical and Recorded Data**: Displays both recorded historical data and historical data for the specified number of days.

Technical Details:

- **API Integration**: Uses the OpenWeatherMap API to fetch weather data.
- **MVC Architecture**: Structured using the Model-View-Controller pattern to separate concerns and improve maintainability.
- **Real-time Updates**: Utilizes SignalR to broadcast weather updates to connected clients in real-time.
- **Caching**: Implements in-memory caching to store frequently accessed data such as coordinates and historical weather data, improving performance.
- **Retry Logic and Timeouts**: Uses Polly for handling HTTP request retries and timeouts to ensure resilience.

Note:

This document provides detailed information about the application's implementation, design patterns, and architecture to ensure a comprehensive understanding of its functionality and structure.

2. Prerequisites

- Visual Studio 2022
- .NET 8 SDK
- Internet connection (for fetching weather data)
- OpenWeatherMap API Key

3. Setup and Installation

Extract the ZIP File :

I. Download the ZIP file containing the solution.

II. Extract the ZIP file to your desired location on your computer.

Setup Instructions :

I. Open Visual Studio.

II. Click on `File` -> `Open` -> `Project/Solution`.

III. Navigate to the extracted directory and select the `WeatherApp.sln` file.

IV. Configure `` `WeatherApiKey` `` in `` `appsettings.JSON` `` with OpenWeatherMap API Key.

V. Restore NuGet packages by right-clicking the solution in Solution Explorer and selecting `` `Restore NuGet Packages` ``.

VI. Click on `Build` -> `Build Solution`.

Note: Currently, there is a OpenWeatherMap API key installed for Application demonstration and evaluation purposes.

Running the Application :

I. Once the build is successful, click on `Debug -> Start Debugging` or press `F5` to run the application with debugging.

II. Alternatively, you can run the application without debugging by clicking on `Debug -> Start Without Debugging` or pressing `Ctrl + F5`.

4. Configuration

Weather Configuration :

Users can set the following configuration options on the main page:

- **Location**: Name of the city to fetch weather data for. It is a mandatory field
- **Update Interval**: Interval (in minutes) for updating weather data. It is a mandatory field.
- **Number of Days for Recorded Historical Data**: Number of past days to fetch the recorded historical weather data by the application in the mentioned location.
- **Number of Days for Historical Data**: Number of days to fetch historical data in the mentioned location.

5. Using the Application

- Open the application in a web browser.
- Enter the desired configuration options in the form provided.
- Click the "**Set Weather Configuration**" button to apply the settings.
- View the current weather details ,recorded historical data by application and historical data on the page.
- The Current weather details of the mentioned city and recorded history will be displayed as it updates for given update interval.

6. Technical Stack

Technologies Used :

- Framework: .NET 8
- Architecture: MVC (Model-View-Controller)
- Programming Languages: C#
- Web API:
    - OpenWeatherMap API : The application uses the OpenWeatherMap API to fetch current weather data and Historical weather data. The API provides comprehensive weather information, including temperature, description, wind speed, and more. To use the API, you need to sign up at OpenWeatherMap and obtain an API key, which is then configured in the WeatherService class to make request to the API.

Libraries and Tools :

- **Newtonsoft.Json:**
  - **Description**: JSON serialization and deserialization.
  - **Installation**: `dotnet add package Newtonsoft.Json`.
- **SignalR**:
  - **Description**: Real-time weather updates.
  - **Installation**: `dotnet add package Microsoft.AspNetCore.SignalR`.

- **Polly**:
  - **Description**: Resilience and transient fault handling (retry logic and timeouts).
  - **Installation**: `dotnet add package Microsoft.Extensions.Http.Polly`.

- **Microsoft.Extensions.Caching.Memory**:
  - **Description**: In-memory caching.
  - **Installation**: `dotnet add package Microsoft.Extensions.Caching.Memory`.

7. Application Structure

I. Models :

WeatherModel : Represents the weather data including current temperature, description, Recorded History and Historical Data.

Weather_History : Manages Historical Records of Recorded weather details.

WeatherData : Represents individual weather records in Weather_History.

WeatherApiResponse : Used for deserializing the data from responses of the API request.

II. Services :

WeatherService: Fetches and Manages weather data from OpenWeatherMap API.

- **Responsibilities**:
  - Fetch current weather data using the OpenWeatherMap API.
  - Retrieve Historical weather data for the configured number of days.
  - Manages Recorded Historical data by the application.
  - Serialize and deserialize JSON responses using Newtonsoft.Json.
  - Implement caching to store coordinates and historical weather data to improve performance.

- Handle HTTP request retries and timeouts using Polly for resilience.

- **Implementation**:
  - Utilizes `HttpClient` for making API calls.
  - Parses JSON responses to extract relevant weather data.
  - Caches frequently accessed data such as coordinates and historical weather data.
  - Uses `Polly` to implement retry logic and timeouts for HTTP requests.
  - Handles potential API errors and logs them using ASP.NET Core's logging framework.

WeatherUpdateService: Background service to periodically update weather data.

- **Responsibilities**:
  - Run as a hosted service that periodically fetches updated weather data.
  - Ensure the weather data is refreshed at the configured interval.
  - Broadcast updated weather data to connected clients using SignalR.
- **Implementation**:
  - Implements 'BackgroundService' for periodic execution.
  - Uses dependency injection to access 'WeatherService' and 'IHubContext<WeatherHub>'.
  - Handles cancellation and proper disposal of tasks to ensure graceful shutdown.

III. Hubs :

WeatherHub: SignalR hub to send weather updates to clients.

- **Responsibilities**:
  - Manage real-time communication between the server and clients.
  - Broadcast updated weather data to all connected clients.
- **Implementation**:
  - Inherits from `Hub` and uses `Clients.All.SendAsync` to push updates. Provides a method 'SendWeatherUpdate' to broadcast weather updates.

IV. Controllers :

WeatherController: Manages weather data configuration and displays current weather.

- **Responsibilities**:
  - Handle user input for configuring weather settings.
  - Fetch and display Current Weather, Historical Data, and Recorded Data .

- Manage validation and error handling for user inputs.

- **Implementation**:
  - Provides an `Index` action to render the main view.
  - Handles form submission via `SetWeatherConfig` action to update weather configuration.
  - Utilizes `WeatherService` to fetch weather data and `WeatherUpdateService` to manage periodic updates.
  - Uses `TempData` to pass success and error messages to the view.

V. Views :

Index View (*Index.cshtml*): The main view where users can configure the weather settings and view the current weather, historical data, and recorded historical data.

- **Responsibilities**:
  - Display forms to input location, update interval, and number of days for historical and Recorded data.
  - Show current weather details including temperature, description, and their Time of Retrieval.
  - Provide tables to display historical data and recorded data.
  - Include client-side validation and real-time updates using SignalR.
- **Implementation**:
  - Uses Razor syntax to bind form inputs and display weather data.
  - Includes jQuery for client-side validation and interaction.
  - Integrates SignalR for real-time updates to dynamically update weather data on the page.
  - Uses Bootstrap CSS for Styling of HTML pages.