Dean H - Live Weather and Map Application Design Document

Introduction

The DeanH_WeatherApp is a .NET and ASP.NET MVC program designed to provide users with real-time weather information overlaid on interactive maps. By integrating data from the OpenWeatherMap and OpenStreetMap APIs, the application offers a convenient all-in-one tool for accessing weather and location-based data.

Key Features

- Through the utilization of the OpenWeatherMap API, the program fetches up-to-date weather information for specified locations. This includes essential weather parameters such as temperature, humidity, and descriptive weather conditions such as sunny, cloudy, or rainy.

- The application extends its capabilities by querying the OpenWeatherMap API for future weather forecasts, enabling users to plan ahead. It processes this data to calculate details like average temperature, highest temperature, and lowest temperature for the upcoming days.

- Using OpenStreetMap API, the application displays geographical data through interactive maps. Users can explore detailed maps of specific locations, manipulate zoom levels, and access information regarding nearby points of interest, streets, and landmarks.

- By using functionalities of both APIs, the application delivers an immersive experience, seamlessly integrating weather information onto an interactive map interface. Users can effortlessly select a location of interest, view its current weather conditions, and explore the surrounding areas through the intuitive map interface.

Additionally, this repository encompasses a companion project housing a suite of unit tests tailored for the .NET components. With 10 crafted unit tests, this project ensures the robustness, performance, and reliability of diverse aspects of the application's functionality.

System Architecture

The Live Weather And Map Application follows a three-tier architecture, consisting of the presentation layer, the application layer and integration with external services. Each layer has specific responsibilities and interacts with the others to fulfill system requirements.

Presentation Layer

The presentation layer consists of the user interface components responsible for rendering weather information and maps to users. This layer includes web pages, views, and client-side scripts that enable users to interact with the application. The user interface is designed to be intuitive and user-friendly, allowing users to easily navigate through weather data and map displays.

Application Layer

The application layer serves as the intermediary between the presentation layer and the data layer. It contains the controllers, business logic, and service components responsible for processing user requests, fetching data from external APIs, and orchestrating the flow of information within the application. The application layer ensures that weather data retrieved from the OpenWeatherMap API and map data from the OpenStreetMap API are integrated seamlessly and presented to users.

Integration with External Services

It integrates external services (OpenWeatherMap and OpenStreetMap APIs) to retrieve weather and map data. Integration with these APIs involves making HTTP requests to fetch real-time weather updates, weather forecasts, and geographical information. The application interacts with the APIs using RESTful principles, exchanging JSON-formatted data over HTTP protocols. Integration with external services is implemented in a modular and extensible manner, allowing for easy maintenance and future enhancements.

Scalability and Performance Considerations

The system architecture of the application is designed to be scalable and performant, capable of handling increasing loads and user interactions. Horizontal scalability is achieved through load balancing and auto-scaling mechanisms, allowing the application to dynamically allocate resources based on demand. Performance optimizations, such as optimizing API calls are implemented to minimize latency and improve responsiveness. Additionally, asynchronous processing and parallelization techniques are employed to enhance throughput and resource utilization.

Components

The Live Weather App is structured into three main components: Model, View, and Controller.

Model

The Model component defines the structure and data manipulation logic for the application.

NextFiveDayForecastData This model includes various classes representing forecast data for the next five days. It consists of classes such as Main, Weather, Clouds, Wind, Sys, List, City, and Coord.

WeatherData This model represents the current weather data. It includes properties such as Coord, MainData, Weather.

These models are responsible for encapsulating data retrieved from the weather API.

View

The View component, implemented using Razor Pages along with HTML, CSS, and JavaScript provides the user interface for interacting with the weather data.

Index.cshtml This view displays the home page of the application, allowing users to input their location and view current weather information. It also displays a five-day weather forecast and a map with the user's location.

IndexPage.js This JavaScript file contains functions for fetching and displaying current weather and five-day forecast data. It also handles user input validation and updates the UI accordingly.

Controller

The Controller component manages the application logic, including routing, data retrieval from the weather API, and rendering views.

HomeController.cs This controller handles HTTP requests related to weather data. It includes methods for retrieving current weather by location or latitude and longitude as well as methods for fetching a five-day weather forecast. It utilizes HttpClient to communicate with the OpenWeatherMap API and returns JSON responses containing weather data.

Overall, this architecture promotes separation of concerns and modularity facilitating efficient development and maintenance.