Find Weather

Overview

"Find Weather" is an iOS app designed to provide users with up-to-date weather information based on their current location or any location they choose to search for. The app utilises location services and integrates with a weather API to fetch accurate and real-time weather data.

Features

- Current Location Weather:
    - Displays the current weather conditions at the user's location upon app launch.
- Search by Location:
    - Allows users to search for weather information in any location worldwide by providing latitude and longitude.

Architecture

The app is built following the MVVM (Model-View-ViewModel) architecture, promoting modularity and maintainability. Dependency injection is implemented to enhance component integration.

Third-Party Libraries

- Kingfisher:
    - Efficiently loads and caches weather-related images for a visually appealing experience.
- NVActivityIndicatorView:
    - Integrates smooth loading indicators, enhancing the user interface during data retrieval.
- ReachabilitySwift:
    - Monitors network reachability to ensure responsiveness and timely weather updates.

Functionality

- Correctness:
    - Thorough testing ensures accurate weather information retrieval and display.
- Adherence to Best Practices:
    - Follows the MVVM design pattern for structured development.
- Code Quality and Readability:
    - Emphasis on clean, high-quality, and readable code for ease of collaboration.

- Error Handling:
    - Robust error handling mechanisms guarantee a stable user experience.
- UI/UX Design:
    - User-centric design for an intuitive and visually appealing weather app.

Code Comments

- Clear and concise code comments are provided throughout the implementation, explaining key aspects of the codebase for better understanding.

Handling Complicated API Responses

The "Find Weather" app interacts with a weather API that delivers responses in different formats based on the type of information requested. To handle these variations, the app employs a robust model class structure and custom serialization methods.

Model Classes

WeatherInfoModel

Represents the basic weather information structure and is used for the primary response when retrieving weather details.

WeatherInfoModel

Extends the basic `WeatherModel` to incorporate additional details or a different structure when more comprehensive weather information is requested.

ForecastLocationModel

Designed to parse location-specific data obtained from the API when users perform a location search by providing latitude and longitude.

Custom Serialisation

To accommodate the diverse response structures, custom serialization methods have been implemented. These methods intelligently parse the API responses into the corresponding model classes, ensuring accurate representation and efficient data handling.

Concurrent API Calls using `DispatchGroup`

The "Find Weather" app efficiently performs concurrent API calls to enhance user experience and retrieve multiple sets of data simultaneously. This is achieved through the use of `DispatchGroup`, a powerful mechanism in Swift for managing asynchronous tasks.

Implementation Overview

In the primary view controller, a `DispatchGroup` is employed to coordinate the execution of two API calls concurrently:

Current Location Weather API Call:

- Initiates an API request to retrieve the current weather information based on the user's location.
- Utilizes the `WeatherInfoModel` for parsing the response.

Forecast API Call:

- Facilitates a second API request when the user performs a location search by providing latitude and longitude.
- Leverages the `ForecastModel` for parsing the response.

Unified API Handling with `fetchData` Function

The "Find Weather" app optimises API interaction by consolidating the logic for fetching data from different endpoints into a single, versatile `fetchData` function. This function is designed to handle various API requests seamlessly, providing a clean and maintainable approach to data retrieval.

`fetchData` Function Overview

The `fetchData` function serves as a unified entry point for making API requests. It incorporates the necessary parameters and logic to manage different API endpoints, allowing for a concise and modular code structure.

IBDesignable for Custom UI Elements

To enhance the ease of UI customization and previewing within Interface Builder, the "Find Weather" app utilises the `@IBDesignable` attribute for certain custom UI classes. This allows developers to visualise and customise the appearance of these elements directly in the Interface Builder.

Contact Information

For further inquiries or collaboration, please contact:
Kathiravan
8778916536
Kathiravang97@gmail.com