

Weather Forecast BLoC App

A Flutter-based mobile weather application utilizing the BLoC pattern and Clean Architecture principles.

Features

- 🔍 Search and display real-time weather by city name
 - 📈 View 24-hour hourly weather forecast
 - 🌡️ Toggle between Celsius and Fahrenheit
 - 🔄 Fetch data from OpenWeatherMap API
 - 🖋️ Includes unit and widget tests
-

Getting Started

Build & Run Instructions

1. Clone the repository:

```
git clone https://github.com/aungmyopaing890/weather_forecast.git
cd weather_forecast
```

2. Install dependencies:

```
flutter pub get
```

3. Add your OpenWeatherMap API key as shown below.

4. Run the app:

```
flutter run
```

API Key Setup

To use the app, obtain a free API key from OpenWeatherMap:

1. Visit [OpenWeatherMap API Guide](#)
2. Add your key to the following file:

```
// lib/core/master_config.dart
const String apiKey = "<YOUR_API_KEY>";
```

Project Structure

```
lib/  
├── config/           # App routes and endpoints  
├── core/             # Common themes, constants, utilities  
├── features/  
│   ├── common/      # Shared UI components  
│   ├── splash/       # Splash screen UI  
│   └── weather/  
│       ├── data/     # Data sources, models, enums, repositories  
│       ├── domain/   # Entities, use cases, repository contracts  
│       └── presentation/ # UI, widgets, BLoC  
└── main.dart         # Application entry point
```

Clean Architecture

Data Layer

- **Models (O):** Serialize and transform raw API data.
- **Repositories:** Implement domain contracts, abstracting the data source.
- **Data Sources (D):** Fetch data from external APIs or services.

Domain Layer

- **Entities:** Core business models shared across app.
- **Repositories:** Define interfaces for data operations.
- **Use Cases (S):** Contain business logic and execute app-specific actions.

Presentation Layer

- **BLoC/Cubit:** Manage state and business flow.
- **Views:** React to state changes and render UI.

Test-Driven Development (TDD)

- All business logic and UI interactions are tested using unit and widget tests.
- **mockito** is used for mocking dependencies.
- Key tested components:
 - **WeatherBloc, HourlyWeatherBloc**
 - Widgets like **GetWeatherButton**, forecast screens

To run all tests:

```
flutter test
```

Dependency Injection

The app uses `GetIt` for dependency injection, ensuring loosely-coupled and easily testable components.

Future Recommendations

- Add support for weekly weather forecasts
Extend the app to include 7-day forecast data using the OpenWeatherMap API for long-term planning.
 - Integrate weather maps (e.g., radar, precipitation)
Visualize weather patterns with map overlays (e.g., wind, rain, clouds) using services like Mapbox or Leaflet.
 - Improve offline support with local caching
Cache previously fetched weather data to allow basic offline access and reduce redundant API calls.
 - Add localization for multiple languages
Support international users by implementing multi-language support with `flutter_localizations`.
 - Dark mode support based on system theme
Enhance user experience by dynamically switching themes based on the device's system setting.
-