A logo with letters and a letter

AI-generated content may be incorrect.Final Project Documentation

Application Name: Weatheree

Author: Geovanni Bonilla

Date: 10/06/2025

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 1. Project Overview

## 1.1 Problem Statement

Checking the weather is a daily need, but users often face cluttered or ad-heavy apps that complicate access to simple forecasts. This application solves the problem by providing a clean, focused, and responsive interface that fetches live weather data quickly and reliably.

## 1.2 Target Audience

This app is designed for users who want a simple and reliable weather forecast tool, including travelers, daily commuters, and anyone who needs instant access to current weather conditions.

## 1.3 Core Features

● Current Weather Data: Fetches live weather information (temperature, humidity, description, and icons).  
● Search by City: Users can search for weather information in any city worldwide.  
● Multi-Screen Navigation: Includes Home, Details, and Settings screens.  
● State Management with Provider: Efficient management of weather data across multiple screens.  
● Clean UI: Simple, user-friendly, and mobile-responsive design.  
● API Integration: Real-time data fetched from OpenWeatherMap API.  
● Testing: Unit and widget tests included.

# 2. Technical Design & Architecture

## 2.1 State Management Strategy

The application uses the Provider package to manage state across multiple screens. Provider was chosen because it is lightweight, easy to integrate, and avoids prop-drilling while keeping the app scalable. Local UI interactions (such as loading states) are managed within stateful widgets.

## 2.2 Data Model

The core model is the Weather class, which stores all information retrieved from the API.

// lib/models/weather.dart  
class Weather {  
 final String cityName;  
 final double temperature;  
 final String description;  
 final int humidity;  
 final double windSpeed;  
 final String icon;  
  
 Weather({  
 required this.cityName,  
 required this.temperature,  
 required this.description,  
 required this.humidity,  
 required this.windSpeed,  
 required this.icon,  
 });  
  
 factory Weather.fromJson(Map<String, dynamic> json) {  
 return Weather(  
 cityName: json['name'],  
 temperature: json['main']['temp'].toDouble(),  
 description: json['weather'][0]['description'],  
 humidity: json['main']['humidity'],  
 windSpeed: json['wind']['speed'].toDouble(),  
 icon: json['weather'][0]['icon'],  
 );  
 }  
}

## 2.3 Persistence / API Strategy

Persistence: No local storage is used; the app fetches data on demand.  
API Strategy: Weather data is fetched from OpenWeatherMap API.  
Example endpoint:  
https://api.openweathermap.org/data/2.5/weather?q={cityName}&appid={API\_KEY}&units=metric

## 2.4 Widget Tree Diagram

Below is a simplified widget tree of the Home Screen:  
  
├── AppBar  
 ├── Column  
 │ ├── TextField (Search City)  
 │ ├── WeatherCard (Current Weather)  
 │ └── ElevatedButton (Details Navigation)  
 └── BottomNavigationBar

# 3. Setup & Installation

## 3.1 Prerequisites

● Flutter SDK (>=3.0.0)  
● Dart (>=3.0.0)  
● Android Studio / VS Code with Flutter extension  
● Git

## 3.2 Installation Steps

1. Clone the repository:  
 git clone https://github.com/Sadexx79/Weatheree.git  
 cd weatheree  
  
2. Install dependencies:  
 flutter pub get  
  
3. Obtain an API key:  
 - Register at OpenWeatherMap (https://openweathermap.org/api)  
 - Copy your API key.  
  
4. Add API key:  
 - Open lib/utils/constants.dart  
 - Replace 'YOUR\_API\_KEY' with your actual key:  
  
 static const String apiKey = 'YOUR\_API\_KEY';  
  
  
5. Run the application:  
 flutter run  
  
6. Run tests:  
 flutter test