WEATHER APP BACKEND DOCUMENTATION

Table of Content

Overview:

The Weather App is a backend application that retrieves weather data from an external API, stores it in a Firestore database, and provides CRUD (Create, Read, Update, Delete) operations to manage weather data for different locations.

Features:

Weather Data Retrieval: Fetches weather data from the OpenWeatherMap API based on the user-provided location.
Data Storage: Stores weather data in a Firestore database for persistent storage.
CRUD Operations: Allows users to perform CRUD operations on weather data for different locations.
Error Handling: Implements error handling to manage failures during data retrieval, storage, and manipulation.

Setup:

Prerequisites
Node.js and npm installed on your system.
Firebase project created and Firestore database initialized.
Installation
Clone the repository: git clone https://github.com/janu-riya/WeatherApp-Firebase-TS-CRUD.git

Navigate to the project directory: cd App #cd <project directory>

Install dependencies: npm install
Configure Firebase: Replace the config.json file with your Firebase service account credentials.

Firestore Project Creation and Setup
Create a Firestore Project:
Go to the Firebase Console.
Click on "Add project" and follow the prompts to create a new project.
Choose a project name and click "Create project".

Enable Firestore Database:
Once the project is created, click on "Firestore Database" from the left sidebar menu.
Click on "Create database" to start the setup process.
Select the location for your database and choose the security rules (Start in test mode is recommended for development).
Click "Next" and then "Done" to complete the setup.
Get Firebase Configuration:
Click on the gear icon next to "Project Overview" and select "Project settings".
Navigate to the "General" tab and scroll down to the "Your apps" section.
Click on the Firebase SDK snippet and select "Config".
Copy the Firebase config object.
Create a Service Account:
Go to "Project settings" in the Firebase Console.
Navigate to the "Service accounts" tab.
Click on "Generate new private key" to download the service account JSON file.
Replace the Config and Service Account:
Replace the configuration in your project's config.json file with the Firebase config object you copied earlier.
Replace the config.json file in your project with the downloaded service account JSON file.

Usage:

Adding Weather Data:
Use the addWeather method to add weather data for a specific location.
Retrieving Weather Data:
Use the getWeather method to retrieve weather data for a specific location.
Updating Weather Data:
Use the updateWeather method to update weather data for a specific location.
Deleting Weather Data:
Use the deleteWeather method to delete weather data for a specific location.

Code Structure:

The project structure consists of the following main components:

src/models: Contains the WeatherModel interface to define the structure of weather data.
src/services: Includes the FirestoreService class responsible for interacting with Firestore.
src/index.ts: Entry point of the application where CRUD operations are demonstrated.

Dependencies:

firebase: Firebase SDK for accessing Firebase services.
axios: Promise-based HTTP client for making requests to external APIs.
@types/node: TypeScript type definitions for Node.js.

Configuration:

Ensure the following configurations are correctly set up:
Firebase configuration in config.json.
OpenWeatherMap API key in the getWeather function.

Error Handling:

The application handles errors gracefully by logging them to the console and throwing exceptions when necessary.

Future Enhancements:

Implement user authentication and authorization.
Add support for caching weather data to improve performance.
Expand CRUD operations to support bulk operations and filtering.