API DESIGN DOCUMENTATION - Distributed Weather App

1. Overview

The **Distributed Weather API** provides real-time and historical weather data to clients. It is designed for **high availability, scalability, and low latency**, serving millions of requests daily.

- **Tech Stack:** RESTful API, JSON-based responses
- **Data Sources:** Internal weather stations, external APIs (e.g., OpenWeather, NOAA)
- **Caching Strategy:** Redis for hot data (frequent locations), CDN for static responses
- **Rate Limiting:** Token-based rate limiting to prevent abuse

2. Base URL

- **Production:** `https://api.weatherapp.com/v1/`
- **Staging:** `https://staging.weatherapp.com/v1/`

3. Authentication

- **API Key-based authentication** (`x-api-key` in headers)
- **OAuth 2.0 support** for third-party integrations
- **JWT-based token authentication** for user-specific requests

Example Request:

```
GET /weather?lat=40.7128&lon=-74.0060
Headers:
  x-api-key: YOUR_API_KEY
```

4. Endpoints

Endpoint	Method	Description	Auth Required?	Rate Limited?
`/weather`	`GET`	Get real-time weather for a location	Yes	Yes
`/forecast`	`GET`	Get 7-day weather forecast	Yes	Yes
`/history`	`GET`	Fetch historical weather data	Yes	Yes
`/alerts`	`GET`	Get active weather alerts	Yes	Yes
`/stations`	`GET`	Get weather station metadata	No	Yes
`/user/preferences`	`POST`	Save user weather preferences	Yes	No

5. API Request & Response Details

1. Get Real-time Weather

- **Endpoint:** `/weather`
- **Method:** `GET`
- **Query Params:**
  - `lat` (float) – Latitude
  - `lon` (float) – Longitude
  - `units` (string) – `metric` | `imperial` (default: `metric`)
- **Response:**

```
{
  "location": "New York, NY",
  "coordinates": { "lat": 40.7128, "lon": -74.0060 },
  "temperature": { "value": 22.5, "unit": "C" },
  "humidity": 70,
  "wind_speed": { "value": 15, "unit": "km/h" },
  "condition": "Cloudy",
  "timestamp": "2025-03-21T10:00:00Z"
}
```

2. Get 7-day Forecast

- **Endpoint:** `/forecast`
- **Method:** `GET`
- **Query Params:** `lat`, `lon`, `units`
- **Response:**

```
{
  "location": "New York, NY",
  "coordinates": { "lat": 40.7128, "lon": -74.0060 },
  "forecast": [
    { "date": "2025-03-22", "temperature": { "min": 18, "max": 24 },
"condition": "Sunny" },
    { "date": "2025-03-23", "temperature": { "min": 16, "max": 22 },
"condition": "Rainy" }
  ]
}
```

3. Get Historical Weather Data

- **Endpoint:** `/history`
- **Method:** `GET`
- **Query Params:**
  - `lat`, `lon`
  - `start_date` (YYYY-MM-DD)
  - `end_date` (YYYY-MM-DD)
- **Response:**

```
{
  "location": "New York, NY",
  "coordinates": { "lat": 40.7128, "lon": -74.0060 },
```

```
  "data": [
    { "date": "2025-03-15", "temperature": { "avg": 20 }, "condition":
"Cloudy" },
    { "date": "2025-03-16", "temperature": { "avg": 18 }, "condition":
"Rainy" }
  ]
}
```

4. Get Weather Alerts

- **Endpoint:** `/alerts`
- **Method:** `GET`
- **Query Params:** `lat`, `lon`
- **Response:**

```
{
  "location": "New York, NY",
  "alerts": [
    { "type": "Storm Warning", "severity": "High", "issued_at": "2025-03-
21T06:00:00Z" }
  ]
}
```

6. Rate Limiting

- **Basic users:** 1000 requests per day
- **Premium users:** 10,000 requests per day
- **Implementation:** Token Bucket Algorithm

Response Header for Rate Limiting:

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 980
X-RateLimit-Reset: 3600
```

7. Caching Strategy

- **Redis**: Cache frequently requested weather data for 5 minutes
- **CDN**: Cache API responses for static requests (e.g., station metadata)
- **Client-side caching**: `Cache-Control: max-age=300`

8. Security & Best Practices

API Gateway (e.g., AWS API Gateway, Kong, Nginx)
DDoS Protection (Cloudflare, AWS Shield)
Data Encryption (TLS 1.3 for HTTPS)
Input Validation (Prevent SQL Injection, XSS attacks)
OAuth 2.0 for third-party integrations