Software Requirements Specification (SRS) for Historical Weather API

Version	Date	Author	Changes
0.1	2025-10-11	Mohamed Abdul	Initial Draft
1.0	2025-11-01	Mohamed Abdul	Final Draft

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to specify the software requirements for the **Historical Weather API**. This product will be a Spring Boot application that provides a simplified, focused API for meteorological researchers. It will act as a facade, consuming the public Open-Meteo API to provide specific data points (min, max, and average temperature) based on user-provided coordinates.

1.2. Document Conventions

This document follows the IEEE 830 standard for SRS documents. Requirement identifiers are sequential, prefixed with "REQ-".

1.3. Intended Audience and Reading Suggestions

- **Meteorological Researchers:** The primary end-users who will consume this API for their research. They should focus on Section 3 to understand the available data.
- **Project Developers (Spring Boot):** Will use this document as the "source of truth" for implementation. All sections are relevant.
- **Project Manager:** To define and manage project scope. Sections 1.4, 2.2, and 3 are most relevant.
- **QA/Testers:** To develop test plans and cases. Sections 3, 4, and 5 are most relevant.

1.4. Project Scope

This project aims to accelerate meteorological research by replacing the time-consuming manual process of finding historical weather data. It does this by providing a simple, stable API for specific, high-demand data points.

The scope of this project includes:

- A RESTful API built using Spring Boot.
- Endpoints for min, max, and average temperature.
- Accepting latitude and longitude as primary inputs.
- Fetching data from the external Open-Meteo API.
- Returning data in JSON format.

The scope does not include:

- A graphical user interface (GUI).
- Weather forecasting or real-time weather data.
- Hosting or storing any historical weather data locally. The API will be a stateless pass-through facade.
- User authentication or account management (V1.0).

1.5. Definitions, Acronyms, and Abbreviations

Term	Definition
API	Application Programming Interface
SRS	Software Requirements Specification
JSON	JavaScript Object Notation
CSV	Comma-Separated Values
REST	REpresentational State Transfer
Spring Boot	A Java-based framework for creating microservices
Facade	An API that simplifies a more complex underlying API
Open-Meteo	The external, public API used as the data source
GET	An HTTP method for retrieving data

1.6. References

ID	Document
$$Ref-1$$	Open-Meteo API Documentation
	https://open-meteo.com/en/docs
$$Ref-2$$	Spring Boot Documentation
	https://spring.io/projects/spring-boot

1.7. Overview

Section 2 provides an overall description of the product and its constraints. Section 3 details the specific functional requirements for each API endpoint. Section 4 describes the external interfaces (specifically the Open-Meteo API). Section 5 lists the non-functional requirements

such as performance and security.

2. Overall Description

2.1. Product Perspective

The Historical Weather API is a new, self-contained microservice. It is part of no larger system but is dependent on the public Open-Meteo API

$$Ref-1$$
for all its data. It serves as a simplifying wrapper, abstracting the complexity of the Open-Meteo API into three simple endpoints.

2.2. Product Features

The system shall provide the following high-level features:

1. **Retrieve Minimum Temperature:** Get the minimum temperature for a given location and date range.
2. **Retrieve Maximum Temperature:** Get the maximum temperature for a given location and date range.
3. **Retrieve Average Temperature:** Get the average temperature for a given location and date range.

2.3. User Classes and Characteristics

Class	Description
Researcher (User)	The primary user. Has a background in meteorology or data science. Is comfortable consuming REST APIs in scripts (e.g., Python, R) or other applications.
Developer (User)	A secondary user. Will integrate this API into a larger application. Requires clear API documentation and reliable error codes.

2.4. Operating Environment

- The system shall be a Java Spring Boot application packaged as a .jar file.
- It shall be deployable in any environment that supports Java Runtime Environment (JRE) 17 or newer (e.g., cloud server, on-premise, container).
- The operating environment must have stable, outbound internet access to

https://api.open-meteo.com on port 443 (HTTPS).

2.5. Design and Implementation Constraints

1. The system **must** be built using the Spring Boot framework.
2. The system **must** fetch all weather data from the https://open-meteo.com API.
3. The system **must not** persist any weather data to a database.
4. All API responses **must** be in JSON format.
5. All API endpoints **must** be exposed over HTTPS.

2.6. User Documentation

- API documentation (e.g., Swagger or OpenAPI) shall be generated and exposed by the Spring Boot application.
- A README.md file will be provided with examples for each endpoint.

2.7. Assumptions and Dependencies

1. **Dependency:** The system is entirely dependent on the availability and performance of the Open-Meteo API.
2. **Assumption:** The Open-Meteo API's contract (endpoints and data structure) will remain stable.
3. **Assumption:** Users will provide valid WGS84 coordinates (latitude and longitude).
4. **Assumption:** Users will provide valid date ranges (start_date and end_date).

3. System Features

This section details the functional requirements for each endpoint.

3.1. System Feature 1: Request Minimum Temperature

- **Endpoint:** GET /api/v1/weather/min
- **Description:** This feature allows a user to retrieve the minimum temperature for a specific location and date range.
- **Priority:** High

Functional Requirements:

ID	Requirement
REQ-3.1.1	The system shall accept the following query parameters: latitude (float, required), longitude (float, required), start_date (string, required, YYYY-MM-DD), end_date (string, required, YYYY-MM-DD).

REQ-3.1.2	The system shall validate the input parameters. If any are missing or malformed, it shall return an HTTP 400 Bad Request response with a clear error message.
REQ-3.1.3	The system shall construct a GET request to the Open-Meteo API (https://api.open-meteo.com/v1/forecast) using the provided parameters and requesting the temperature_2m_min daily variable.
REQ-3.1.4	The system shall parse the JSON response from Open-Meteo.
REQ-3.1.5	The system shall return an HTTP 200 OK response with a JSON body containing the location, date range, and the requested minimum temperature data.
REQ-3.1.6	If the Open-Meteo API is unreachable or returns an error, the system shall return an HTTP 502 Bad Gateway response.

3.2. System Feature 2: Request Maximum Temperature

- **Endpoint:** GET /api/v1/weather/max
- **Description:** This feature allows a user to retrieve the maximum temperature for a specific location and date range.
- **Priority:** High

Functional Requirements:

ID	Requirement
REQ-3.2.1	The system shall accept the following query parameters: latitude (float, required), longitude (float, required), start_date (string, required, YYYY-MM-DD), end_date (string, required, YYYY-MM-DD).

REQ-3.2.2	The system shall validate the input parameters. If any are missing or malformed, it shall return an HTTP 400 Bad Request response with a clear error message.
REQ-3.2.3	The system shall construct a GET request to the Open-Meteo API (https://api.open-meteo.com/v1/forecast) using the provided parameters and requesting the temperature_2m_max daily variable.
REQ-3.2.4	The system shall parse the JSON response from Open-Meteo.
REQ-3.2.5	The system shall return an HTTP 200 OK response with a JSON body containing the location, date range, and the requested maximum temperature data.
REQ-3.2.6	If the Open-Meteo API is unreachable or returns an error, the system shall return an HTTP 502 Bad Gateway response.

3.3. System Feature 3: Request Average Temperature

- **Endpoint:** GET /api/v1/weather/avg
- **Description:** This feature allows a user to retrieve the average temperature for a specific location and date range.
- **Priority:** High

Functional Requirements:

ID	Requirement
REQ-3.3.1	The system shall accept the following query parameters: latitude (float, required), longitude (float, required), start_date (string, required, YYYY-MM-DD), end_date (string, required, YYYY-MM-DD).

REQ-3.3.2	The system shall validate the input parameters. If any are missing or malformed, it shall return an HTTP 400 Bad Request response with a clear error message.
REQ-3.3.3	The system shall construct a GET request to the Open-Meteo API (https://api.open-meteo.com/v1/forecast) using the provided parameters and requesting the temperature_2m_mean daily variable.
REQ-3.3.4	The system shall parse the JSON response from Open-Meteo.
REQ-3.3.5	The system shall return an HTTP 200 OK response with a JSON body containing the location, date range, and the requested average temperature data.
REQ-3.3.6	If the Open-Meteo API is unreachable or returns an error, the system shall return an HTTP 502 Bad Gateway response.

4. External Interface Requirements

4.1. User Interfaces

There are no graphical user interfaces for this product. The only interface is the REST API defined in Section 3.

4.2. Hardware Interfaces

There are no hardware interfaces for this product.

4.3. Software Interfaces

1. **Inbound: Historical Weather API (This Product)**
    - **Interface:** RESTful API
    - **Protocol:** HTTPS
    - **Data Format:** JSON
    - **Details:** See Section 3 for all endpoints.

2. **Outbound: Open-Meteo API**
    - **Interface:** RESTful API
    - **Protocol:** HTTPS
    - **Data Format:** JSON
    - **Details:** The system will act as a client to the https:api.open-meteo.com server. It will primarily use the /v1/forecast endpoint. It must handle JSON responses and standard HTTP error codes from this external dependency.

4.4. Communications Interfaces

- The system shall communicate over TCP/IP.
- All communication (inbound and outbound) shall be encrypted using HTTPS (SSL/TLS).

5. Non-Functional Requirements

5.1. Performance Requirements

ID	Requirement
NFR-5.1.1	Under normal conditions, all API endpoints shall return a response (or error) within **1500 milliseconds (1.5 seconds)**.
NFR-5.1.2	This performance is dependent on the Open-Meteo API responding within 1000ms.
NFR-5.1.3	The system shall be able to handle at least **50 concurrent requests** without degradation in performance.

5.2. Security Requirements

ID	Requirement
NFR-5.2.1	All API endpoints must use HTTPS. HTTP requests should be rejected or redirected to HTTPS.
NFR-5.2.2	The system shall implement API rate limiting to prevent abuse (e.g., max 100 requests per minute per IP).

NFR-5.2.3	The system shall sanitize all input parameters (latitude, longitude, start_date, end_date) to prevent injection attacks, even though no database is used.
NFR-5.2.4	$$Future Version$$ An API Key shall be required for all requests to the API.

5.3. Software Quality Attributes

Attribute	**Requirement**
Reliability	The API shall have an uptime of 99.9%. It must gracefully handle errors from the external Open-Meteo API and return meaningful error codes (e.g., 502) rather than crashing.
Maintainability	The Spring Boot application code must adhere to standard Java conventions, be well-commented, and include unit tests for all service-layer logic.
Usability	(Applies to API usability) The API endpoints shall be logical and predictable. Error messages must be clear JSON strings, explaining what went wrong (e.g., "Parameter 'latitude' is missing").
Portability	The application shall be packaged as a self-contained .jar file (or Docker container) and be runnable in any environment with the correct JRE, with no other external dependencies besides the Open-Meteo API.

5.4. Business Rules

ID	**Requirement**

BR-1	The date range (end_date - start_date) for a single request cannot exceed 365 days. If it does, a 400 Bad Request error shall be returned.
BR-2	start_date cannot be later than end_date.

Appendix A: Glossary

(See Section 1.5)

Appendix B: Analysis Models

Use Case Diagram

- **Actors:**
  - Researcher
  - Developer
- **System:**
  - Historical Weather API
- **Use Cases:**
  - Get Minimum Temperature
  - Get Maximum Temperature
  - Get Average Temperature
- **Secondary Actor / System:**
  - Open-Meteo API (Provides data for all use cases)

Data Flow Diagram (Context)

- **External Entity (User):** Sends (latitude, longitude, date_range) via an API Request.
- **Process (0 - Historical Weather API):** Receives the request.
  - Sends a (formatted_request) to the Open-Meteo API.
  - Receives (raw_weather_data) from the Open-Meteo API.
  - Processes the raw data.
  - Sends (formatted_min_max_avg_data) to the User.
- **External Entity (Open-Meteo API):** Receives and fulfills data requests.