ClimaTune Developer Guide

1. Introduction and Goals

This document serves as the technical specification and guide for the ClimaTune Prototype, a web-based alarm clock that personalizes music playback based on weather conditions.

Key Objectives:

1. Securely integrate Spotify Web API (OAuth 2.0) for playlist and playback control.
2. Fetch real-time/forecast weather data using the OpenWeatherMap API.
3. Implement a reliable Alarm Scheduler that triggers logic at precise times.
4. Develop a user-friendly frontend (HTML/CSS/JavaScript) for alarm and mapping management.

2. System Architecture Overview

ClimaTune is implemented as a thin, single-page web application client, relying heavily on external APIs.

2.1 Core Components

Component	Description	Technologies
Frontend UI	Handles all user interaction: alarm creation, mapping setup, and settings (theme, volume).	HTML, CSS, JavaScript
Alarm Scheduler	Client-side logic that monitors the system clock and triggers the Alarm Logic function when a scheduled time is reached.	JavaScript setInterval/setTimeout or Web Workers (for better precision).

Spotify Client	Manages OAuth 2.0 authentication, token refresh, playlist retrieval, and music playback via the Spotify Web API.	JavaScript fetch, OAuth 2.0 implementation.
Weather Client	Fetches current or forecasted weather data from the OpenWeatherMap API using the stored location for the triggered alarm.	JavaScript fetch, JSON response handling.
Data Storage	Persistence for user data (alarms, preferences, tokens).	Local Storage (for simplicity of prototype).

3. Data Structures and Data Flow

3.1 Key Data Structures

Data structures are persisted locally (e.g., as JSON objects in Local Storage):

Object Name	Key Fields (Example)	Purpose
Alarm	AlarmID, AlarmTime (HH:MM), AlarmDays (Array of Mon-Sun), Enabled (Boolean), Location, DefaultSong (Fallback identifier).	Defines alarm configuration.
Mapping	WeatherCondition (e.g., "Sunny", "Rain"), SpotifyID (Playlist or Track ID), Priority (High/Low).	Links specific weather conditions to Spotify music.

UserSession	SpotifyAccessToken, SpotifyRefreshToken, TokenExpirationTimestamp, Theme (Light/Dark).	Manages user session and preferences.

3.2 Alarm Activation Logic Flow

The core system function runs when the Alarm Scheduler triggers:

1. Read Alarm: Scheduler identifies an active, enabled alarm for the current time/day.
2. Fetch Weather: Weather Client calls OpenWeatherMap API using the alarm's Location.
    - Data: Retrieves a WeatherCondition object (e.g., {condition_code: 800, description: "Clear Sky"}).
3. Find Match: The system interprets the condition and checks the stored Mapping data.
    - Logic: Prioritizes a specific track mapping over a playlist mapping.
4. Playback:
    - Success: Spotify Client uses the SpotifyAccessToken to initiate playback of the selected track via the Spotify Web API.
    - Failure (API down or No Match): The system falls back to playing the configured DefaultSong (a built-in audio asset).

4. API Integration Details

4.1 Spotify Web API (External System Interface)

- Authentication: Uses OAuth 2.0 Authorization Code Flow.
    - The application must securely handle the retrieval and storage of the Access Token and Refresh Token.
    - The Access Token is required for all data/playback requests and must be checked against its TokenExpirationTimestamp before use.

    - Token Refresh: If expired, the Refresh Token must be used in a background request to obtain a new Access Token without user re-login.
- Endpoints Used (Example):
    - /authorize (Initial user login)
    - /api/token (Token exchange/refresh)
    - /me/playlists (Playlist retrieval for mapping)
    - /me/player/play (Initiate song playback)

4.2 OpenWeatherMap API (External System Interface)

- Data Retrieval: Fetches current or 5-day forecast data. The design uses the weather condition codes (e.g., 800 for clear, 500-531 for rain) to standardize and simplify the mapping logic.
- Error Handling: Implementation includes robust handling for:
    - Network latency/timeouts.
    - API rate-limit constraints (suggested: implement simple caching of weather results for 30 minutes to reduce calls).
    - Invalid API Key or location requests.

5. Implementation and Constraint Management

5.1 Technology Constraints and Mitigation

Constraint/Issue	Impact	Mitigation Strategy
Spotify Token Expiration	Loss of music functionality.	Implement a separate background function to automatically use the Refresh Token before the Access Token expires.

Browser Alarm Precision	JavaScript timers are not precise, especially when the browser tab is inactive or minimized.	Use Web Workers or alternative browser-specific timer APIs (if available) to maintain better clock precision. Advise users to keep the tab active.
Local Storage Limits	Limited capacity and no built-in encryption.	Store only essential user data (tokens, alarm configurations). Avoid storing large datasets. Assume platform-level security for token storage.
Network Reliance	Functionality fails without internet.	The Fallback Alarm Sound is mandatory to ensure reliability under all network conditions.