Requirements Document

WeatherNow Calendar Integration

Pascal Putz
pascal.putz@study.thws.de
5123135

Gunn Kataria
gunn.kataria@study.thws.de
9125072

Katrina Alex
katrina.alex@study.thws.de
9125071

Manuel Stöth
manuel.stoeth@study.thws.de
5123045

Marvin Kraus
marvin.kraus@study.thws.de
5123143

April 2025

1 Author Information

Name	**Email**	**StudyID**
Pascal Putz	pascal.putz@study.thws.de	5123135
Gunn Kataria	gunn.kataria@study.thws.de	9125072
Katrina Alex	katrina.alex@study.thws.de	9125071
Manuel Stöth	manuel.stoeth@study.thws.de	5123045
Marvin Kraus	marvin.kraus@study.thws.de	5123143

2 Project Overview

2.1 Project Description

WeatherNow Calendar Integration is a web-based application designed to simplify free-time planning by combining real-time weather data with the user's personal calendar schedule. By leveraging the OpenWeatherMap API and the Google Calendar API, the app allows users to check current and forecasted weather conditions alongside their availability. Based on weather statistics and free days, WeatherNow suggests personalized activities—whether it's a sunny outdoor hike or a cozy indoor museum visit. The frontend is built using React.js to ensure a modern, responsive, and smooth user experience across both mobile and desktop devices.

2.2 Application Goals

- Enable users to view accurate current weather and detailed forecasts for any location worldwide.
- Integrate the user's Google Calendar to identify free and busy periods automatically.
- Suggest context-aware activities based on weather conditions and user availability.
- Provide a user-friendly, accessible, and visually appealing interface with light/dark mode options.
- Ensure fast, reliable, and error-tolerant performance for all weather and calendar interactions.
- Build a modular and scalable frontend that can be extended with additional APIs or features in the future.

2.3 Target Users

- **Travelers and Tourists** planning outdoor or sightseeing activities.
- **Daily Commuters** preparing for weather disruptions during workdays.
- **Outdoor Enthusiasts** (hikers, cyclists, sports players) relying on forecasts.
- **Busy Professionals and Students** maximizing free time with suitable recommendations.
- **Families and Groups** organizing weekend outings or vacations dependent on weather.

3 Project Complexity

This project exhibits moderate complexity:

- **Multiple Views:** Home, Forecast, Calendar, Settings.
- **API Integration:** Asynchronous calls to two external services.
- **Interactive Widgets:** Calendar component, geolocation lookup.
- **State Management:** React Context (or Redux) for global state.
- **Responsive Design:** Mobile-first layout with CSS Grid/Flexbox.

4 Key Features

Must-Have

- **City Search & Current Weather:** Users search a city, see temperature, wind, humidity, conditions and icon.
- **5-Day Forecast:** Forecast in 3-hour intervals with daily high/low summaries.
- **Geolocation Lookup:** Automatically fetch weather for user's current coordinates.
- **Unit Toggle:** Switch between °C and °F.
- **Theme Toggle:** Light and dark modes, persisted across sessions.
- **Interactive Calendar:** Select any date to trigger analysis.
- **Activity Suggestions:** Recommend indoor/outdoor activities based on weather rules.
- **Error Handling:** Friendly messages for invalid input, denied permissions, or API failures.
- **Responsive UI:** Adapts to mobile, tablet, and desktop breakpoints.
- **Client-Side Navigation:** React Router routes for Home, Forecast, and Calendar.

Nice-to-Have

- Store and recall recently searched cities.
- Sync and overlay multiple Google calendars.
- Localize UI text and weather descriptions (e.g., EN/DE).
- Lookup historical weather for past dates.
- Push notifications for severe weather or upcoming free days.

5 User Roles and Interactions

5.1 Primary User Role

The primary user of the application is the general user—an individual seeking to integrate real-time weather information with their personal calendar to make informed decisions about daily activities. This includes travelers, commuters, students, and outdoor enthusiasts who benefit from location-based and schedule-aware recommendations.

5.2 Key User Interactions

- **Search Weather by City:** Users can input or select a city to view its current weather conditions and forecast.
- **Grant Calendar Access:** Users may authorize the application to access their Google Calendar in order to detect available time slots.
- **Select a Calendar Date:** Clicking on a date in the calendar view fetches weather data for that day and displays personalized activity suggestions.
- **Toggle Display Settings:** Users can switch between temperature units (°C/°F) and choose between light and dark themes for better visual comfort.
- **Navigate Between Views:** Navigation between the Home, Forecast, and Calendar pages is supported through a fixed header or side menu.
- **Receive Feedback and Notifications:** The application provides real-time feedback through inline messages or toast notifications in case of input errors, denied permissions, or API issues.

5.3 Example User Journey

1. A user opens the application on their mobile browser and allows location access.
2. The app fetches and displays the current weather for their location.
3. The user searches for an upcoming weekend date on their calendar.
4. The app retrieves their availability and weather forecast for that date.
5. Based on sunny conditions and free time, the app suggests an outdoor cycling trip.
6. The user switches the theme to dark mode for evening browsing and toggles the temperature to Fahrenheit.
7. When the user tries an invalid city name, the app shows a helpful toast notification: “City not found. Please try again.”

6 User Stories / Use Cases

ID	Title	Description
US1	Search City	As a user, I enter a city name to view current weather. This helps me check the weather anywhere in the world quickly and easily.
US2	View Forecast	As a user, I request a five-day forecast to plan ahead. This allows me to make informed decisions about travel or outdoor activities in the coming days.
US3	Use My Location	As a user, I grant location access to see local weather. This saves time and provides accurate data based on my current GPS location.
US4	Select Date	As a user, I click a calendar date to get weather stats and activity suggestions. This enables me to plan free-time activities according to the forecast.
US5	Toggle Units	As a user, I switch between Celsius and Fahrenheit. This ensures the app respects my regional preferences or personal comfort with temperature units.
US6	Toggle Theme	As a user, I enable dark mode at night. This reduces eye strain and improves usability in low-light environments.
US7	Handle Error	As a user, I get notified on API failures or invalid input. This helps me understand what went wrong and how I can fix it, such as retrying or correcting input.
US8	Recent Searches	As a user, I want to revisit recently searched cities for quick access. This improves efficiency by allowing me to check multiple cities I care about frequently.
US9	Severe Weather Alert	As a user, I want to be notified of severe weather to adjust my plans. This allows me to stay safe and react early to dangerous or disruptive conditions.
US10	Multi-language Support	As a multilingual user, I want to view weather descriptions in my preferred language. This improves accessibility and ensures clarity for non-English users.

7 Non-Functional Requirements

- **Usability:** Clear labels, consistent icons, minimal learning curve.
- **Responsiveness:** Fluid layout using CSS Flexbox/Grid for all viewports.
- **Accessibility:** ARIA labels, sufficient contrast, keyboard navigation.
- **Performance:** Data fetch and render complete under 1.5 s on average networks.
- **Security:** HTTPS only; API keys in environment variables.
- **Maintainability:** Modular React components; documented code.
- **Scalability:** Architecture supports adding new APIs/features with minimal refactoring.

8 Technology Assumptions

- **Frontend:** React.js (ES6+).
- **Styling:** CSS Modules or Tailwind CSS.
- **Routing:** React Router DOM.
- **APIs:** OpenWeatherMap REST, Google Calendar OAuth2.
- **State Management:** React Context or Redux.
- **Build/Deploy:** Docker multi-stage build; Nginx serves static files.

9 Project Constraints

- **OpenWeatherMap API Limit:** The app uses the student tier of OpenWeatherMap, which allows up to 3,000 API calls per minute. Efficient usage and caching are required to stay within this limit.
- **Google Calendar Quotas & OAuth:** Calendar integration is subject to Google's API quotas and OAuth consent requirements. Users must authorize access, and rate limits must be respected.
- **Permission Dependency:** The app relies on user permission for location and calendar access. If denied, some features (e.g., weather-based suggestions) won't function and must be handled gracefully.
- **Frontend-Only Architecture:** No custom backend will be developed. All logic, state management, and API calls are handled in the frontend, which limits secure data handling and processing.
- **Submission Deadline:** The Requirements Document is due on April 30, 2025, at 23:59, and sets the foundation for all following development phases.

10 Acknowledgement of AI Tools

This document was drafted and refined using GPT-4o based on an outline provided by the authors. The authors reviewed and enhanced the output for coherence and alignment with course requirements.