WeatherGuard Attendance

Team: 02

Team members:

Nat Grimm , Josh Clemens , Johnny Huynh, Roger Karam

Software Requirements Specification Document

Version: v2.0.0 **Date: (12/05/2025)**

Table of Contents

Change Log

– Section 2: Updated UI framework from JavaSwing to JavaFX, replaced CSV file storage with MongoDB database through document.

– Section 4: Updated interface requirements to not include detailed UI design.

– Section 6: Add unique requirement identifiers for each attribute.

1 Purpose

Educational institutions in extreme weather locations such as Alaska, northern Canada, and Scandinavian countries face unique challenges in managing student attendance during severe weather conditions. Traditional attendance methods like roll call or paper-based systems become particularly problematic when weather conditions affect student transportation, create safety concerns, or require rapid decision-making about school closures.

Current attendance systems fail to account for weather-related factors that significantly impact student presence. During blizzards, ice storms, or extreme cold warnings, schools often operate with reduced attendance but lack efficient ways to track which students are present and adjust expectations accordingly. Teachers waste valuable instructional time on manual roll call while students arrive throughout the day due to weather-related delays.

A new system is needed because existing attendance solutions do not consider meteorological factors that are critical for institutions in extreme weather regions. The current methods waste time, create safety risks when students travel in dangerous conditions, and provide no data to support weather-related attendance policies. Schools need a system that combines real-time weather monitoring with efficient QR code-based attendance tracking, allowing for quick student check-ins while providing administrators with weather-contextual attendance data to make informed decisions about school operations and student safety.

2 Scope

WeatherGuard Attendance is a comprehensive desktop application that integrates real-time weather monitoring with QR code-based attendance tracking specifically designed for educational institutions in extreme weather regions. The application will be implemented as a Java desktop application utilizing JavaFX for the graphical user interface, weather APIs for current conditions and forecasts, and MongoDB database for attendance data persistence. The system will display current weather conditions and 5-day forecasts alongside attendance data, enabling administrators to make informed decisions about school operations and attendance policies.

Students will check in using unique QR codes generated by the system, allowing for rapid, contactless attendance tracking even when students arrive at different times due to weather delays. The application will integrate with OpenWeatherMap API to retrieve real-time weather data and forecasts, automatically recording weather conditions at the time of each attendance check-in and providing real-time attendance dashboards that show both individual student status and overall attendance patterns correlated with weather conditions.

Key benefits include reducing attendance-taking time by 80%, improving student safety through weather-aware attendance policies, providing data-driven insights for school closure decisions, and ensuring accurate attendance records that account for weather-related absences. The system will eliminate the need for manual roll call, and provide valuable analytics for improving school operations in extreme weather conditions. Schools can use this data to optimize transportation routes, adjust staffing levels, and communicate more effectively with parents about weather-related schedule changes.

3 User characteristics

WeatherGuard Attendance will serve educational institutions in extreme weather regions, including schools in Alaska, northern territories, and other locations where weather significantly impacts daily operations.

3.1 Key users

Teachers and administrative staff who are responsible for tracking student attendance and making weather-related operational decisions.

User role responsibilities: Monitor student attendance, assess weather conditions for safety decisions, generate attendance reports that account for weather factors, and communicate with parents about weather-related attendance policies.

Subject matter experience: Journeyman to Master. Users have extensive experience with educational systems and student management but may lack technical expertise in weather data interpretation and QR code systems.

Technological experience: Novice to Journeyman. Many educators are comfortable with basic technology but need intuitive interfaces. The system must work reliably on desktop computers that teachers commonly use.

Other user characteristics: Education professionals who prioritize student safety and learning continuity. Often work in resource-constrained environments and need solutions that are reliable in extreme weather conditions. Age range typically 25-60, with varying comfort levels with technology but high motivation to ensure student safety and accurate record-keeping.

3.2 Secondary users

School administrators, parents, and transportation coordinators who need weather-contextual attendance information for decision-making.

User role responsibilities: Make school closure decisions, coordinate transportation adjustments, communicate with families about weather-related policies, and analyze attendance patterns for policy development.

Subject matter experience: Master. Have extensive experience with school operations and understand the critical relationship between weather conditions and student attendance in extreme climates.

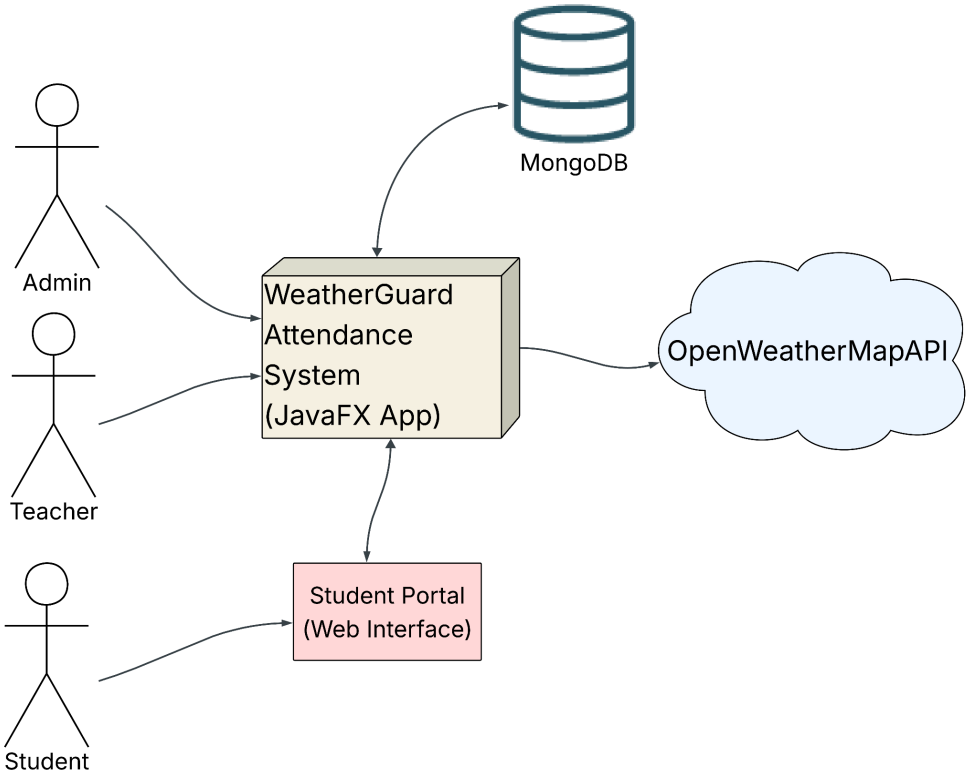
Technological experience: Journeyman. Comfortable with administrative software systems and can interpret data dashboards and reports for decision-making purposes.

Other user characteristics: Decision-makers who must balance educational continuity with student safety. Need access to real-time data and historical trends to make rapid decisions about school operations. Require desktop access for administrative functions and the ability to communicate decisions quickly to multiple stakeholders. Often work in emergency management roles during severe weather events and need reliable, weather-resistant communication tools.

4 Product perspective

4.1 System Context

WeatherGuard Attendance operates as a standalone desktop application that interfaces with external weather services and manages local data storage. The system acts as the central hub for attendance tracking and weather monitoring.



4.2 User interfaces

The system shall provide a JavaFX-based graphical user interface optimized for desktop use.

1) Administrator View:

- The system shall provide an administrative interface for managing classes and rosters.
- The system shall allow administrators to upload class rosters from CSV files.
- The system shall allow administrators to add, edit, and delete class information.
- The system shall display a list of all active classes with relevant metadata.
- The system shall provide navigation to open individual class sessions.

2) Class View (Teacher Interface)

- The system shall provide a teacher interface displaying real-time attendance status for all students in a class.

- The system shall display student attendance with visual indicators (color-coded status).

- The system shall display current weather conditions and 5-day forecasts.

- The system shall generate and display QR codes for student check-in sessions.

- The system shall provide controls to start and end attendance sessions.

- The system shall display attendance statistics in graphical format (pie charts).

3) Student Check-In View:

- The system shall provide a web-based interface accessible via QR code scanning.

- The system shall validate student enrollment before recording attendance.

- The system shall prevent duplicate check-ins for the same session.

- The system shall provide immediate feedback upon successful check-in.

General UI Requirements

- The system shall display error messages with clear descriptions and suggested actions.
- The system shall provide visual feedback for all user actions (button animations, loading indicators).
- The system shall use consistent visual design across all views.
- The system shall support window resizing with appropriate content scaling.

4.3 Software interfaces

OpenWeatherMap API

- Source: https://openweathermap.org/api
- Communication method: HTTPS REST requests
- Data format: JSON responses
- Interface specification: System shall send GET requests with location parameters and receive JSON responses containing temperature, weather conditions, alerts, and forecast data. Authentication via API key.

ZXing (Zebra Crossing) QR Code Library

- Source: https://github.com/zxing/zxing
- Purpose: QR code generation for class check-in sessions
- Interface specification: Java library imported as dependency for generating QR code images

Student Check-In Portal

- Interface specification: System shall integrate with existing school web infrastructure for student authentication and check-in selection. Implementation details managed by the school IT department. For project demonstration, this interface will be simulated.

4.4 Deployment requirements

Java Runtime Environment:

- Java Runtime Environment (JRE) version 11 or higher must be installed on the computer.
- Active internet connection required for weather API functionality
- Minimum 1 Mbps recommended for API calls

Operating System:

- Compatible with Windows 10/11, or macOS 10.14+,

Installation:

- The system shall be distributed as an executable JAR file.
- No additional installation steps required beyond Java runtime.

5 Assumptions and Dependencies

Assumptions:

- Users have basic computer literacy and can navigate desktop applications
- Schools have reliable internet connectivity during operating hours
- The school's location remains constant (weather data for a single geographic area)
- Students have access to mobile devices capable of scanning QR codes

Dependencies:

- OpenWeatherMap API availability: System depends on continuous availability of OpenWeatherMap API services. If the API is unavailable, weather features will be degraded but attendance tracking will continue to function.
- Java Runtime Environment: System requires JRE 21+ to be installed and maintained on deployment machines.
- Internet connectivity: Real-time weather updates depend on active internet connection. The system shall cache last known weather data if connection is temporarily lost.

6 Specific requirements

6.1 System Functional Requirements

R.1 Teacher QR Attendance Check-In

> User Story: As a teacher, I want students to check in with a unique QR code, so that I can take attendance quickly and reduce class time lost to take roll.
>
> Detailed Description:

The system shall generate a unique QR code for each class session that encodes a URL to the attendance check-in interface. When a student scans the QR code, the system shall present a list of students enrolled in that class. When a student selects their name from the list, the system shall record their attendance with the current timestamp. The system shall automatically retrieve and store current weather conditions at the moment of check-in using the OpenWeatherMap API. The system shall display real-time updates on the teacher's dashboard as students check in. The system shall prevent duplicate check-ins for the same student in the same class session. The system shall allow teachers to manually mark students present or absent if needed.

Acceptance Criteria:

When a student comes to class, when they scan the QR code, then their attendance is marked in the system with current weather data. Given the QR scan is successful and the teacher views attendance, then the student is shown as present. Given 30 students check in, when viewing the dashboard, then all 30 students appear as present within 2 seconds of their check-in. Priority: High. Story Points: 5.

R.2 Administrator Weather Dashboard

User Story: As an administrator, I want weather conditions displayed in real time alongside attendance, so that I can make informed decisions about closures and safety policies.

Detailed Description:

The system shall retrieve current weather data from OpenWeatherMap API including temperature, conditions, wind speed, and any active weather alerts. The system shall display current weather conditions in a dedicated panel on the main dashboard. The system shall retrieve and display a 5-day weather forecast. The system shall automatically refresh weather data every 15 minutes. The system shall display attendance statistics alongside current weather information on the same screen. When severe weather alerts are active (as indicated by OpenWeatherMap API), the system shall highlight these alerts prominently with visual indicators. The system shall allow administrators to view historical attendance data correlated with weather conditions for any selected date range.

Acceptance Criteria:

Given weather data is retrieved from the API, when an administrator views the dashboard, then both current weather and attendance information is displayed. Given severe weather alerts exist, when the system updates, then the dashboard highlights risks with red warning indicators, Given the application is running, when 15 minutes have elapsed, then weather data is automatically refreshed. Priority: High, Story Points: 8.

6.2 Logical Database Requirements

The system shall store all data in the MongoDB database. MongoDB is a NoSQL document database that stores data in JSON-like documents.

- **DB-REQ-1:** MongoDB Connection: The system shall connect to MongoDB using a connection string configured in config.properties. The system shall use the database name specified in config.properties. The system shall implement a singleton pattern for the database manager to ensure a single connection instance.
- **DB-REQ-2:** Classes Collection: Document Structure: "_id": ObjectId, "classId": String (required, indexed for fast lookup),"className": String (required),"semester": String,"year": Integer, "startDate": String,"endDate": String,"professorName": String (required),"city": String (required, used for weather lookup), "active": Boolean (required, default: true).
- **DB-REQ-3**: Students Collection: Document Structure: "_id": ObjectId,"studentId": String (required, indexed),"studentName": String (required), "classId": String (required, foreign reference to classes.classId).
- **DB-REQ-4:** Sessions Collection: Document Structure:"_id": ObjectId, "classId": String (required, foreign reference),"sessionId": String (required, unique, timestamp-based YYYYMMDD_HHMMSS),"createdAt": String (ISO 8601 format),"weatherData": String (JSON string of weather conditions), "active": Boolean (required, indicates if session is currently active).
- **DB-REQ-5:** Attendance Collection: Document Structure:"_id": ObjectId, "classId": String (required, foreign reference),"sessionId": String (required, foreign reference), "studentId": String (required, foreign reference), "studentName": String (required),"checkInTime": String (ISO 8601 format), "status": String (required, default: "present").
- **DB-REQ-6:** Data Access Operations: The system shall implement the following database operations:
    - Create new documents (insertOne, insertMany)
    - Read documents with filtering (find, findOne)
    - Update documents (updateOne, replaceOne)
    - Soft delete (update active flag, do not physically delete)
    - Query with filters for efficient data retrieval

6.3 Software System Attributes

6.3.1 Usability

- **US-REQ-1:** First-time users with basic computer skills shall be able to perform attendance check-in within 5 minutes of instruction.
- **US-REQ-2:** Error messages shall be displayed in clear, non-technical language with specific guidance on how to resolve the issue.
- **US-REQ-3:** The system shall use consistent color coding: green for present, red for absent, yellow for severe weather alerts.
- **US-REQ-4:** Font sizes shall be minimum 12pt for body text and 14pt for headers to ensure readability.
- **US-REQ-5:** All user actions shall provide immediate visual feedback (e.g., button press animation, loading indicators).

6.3.2 Performance

- **PERF-REQ-1:** The system shall retrieve weather data from OpenWeatherMap API and display results within 3 seconds under normal network conditions.
- **PERF-REQ-2:** The system shall support simultaneous check-in of 50 students within a 5-minute window without performance degradation.

- **PERF-REQ-3:** The application startup time shall not exceed 5 seconds on systems meeting minimum hardware requirements.
- **PERF-REQ-4:** Weather data refresh operations shall occur in a background thread and not freeze the user interface.

6.3.3 Reliability/Dependability

- **REL-REQ-1:** In the event of OpenWeatherMap API failure, the system shall continue to function using cached weather data and display a warning indicator.
- **REL-REQ-2:** The system shall cache the most recent weather data locally and use this data if API is unreachable for up to 1 hour.

6.3.4 Security

- **SEC-REQ-1:** The system shall store the OpenWeatherMap API key in a configuration file separate from the application code.
- **SEC-REQ-2:** The system shall use HTTPS for all communication with OpenWeatherMap API to ensure data is encrypted in transit.
- **SEC-REQ-3:** The system shall restrict file system access to only the designated application data directory.
- **SEC-REQ-4:** The system shall not store or transmit student passwords (authentication is handled by the school's existing systems).

6.3.5 Maintainability

- **MAIN-REQ-1:** The system shall use a modular architecture with separate packages for UI, data access, weather service integration, and business logic.
- **MAIN-REQ-2:** All classes shall follow Java naming conventions and include Javadoc comments describing purpose, parameters, and return values.
- **MAIN-REQ-3:** The system shall use meaningful variable names that are self-documenting.
- **MAIN-REQ-4:** Configuration values (API endpoints, file paths, refresh intervals) shall be externalized to a properties file for easy modification without code changes.
- **MAIN-REQ-5:** The system shall implement logging of all errors and significant events to a log file for troubleshooting purposes.
- **MAIN-REQ-6:** CSV file format version shall be included in file headers to support future format changes without breaking existing data.