Deliverable #1 Template : Software Requirement Specification (SRS)

SE 3A04: Software Design II – Large System Design

Tutorial Number: T01
Group Number: G4
Group Members: Lukas Buehlmann, David Olejniczak,Vanessa Lai, Saqib Khan, Suzanne Abdullah

IMPORTANT NOTES

- Be sure to include all sections of the template in your document regardless whether you have something to write for each or not
  - If you do not have anything to write in a section, indicate this by the *N/A*, *void*, *none*, etc.
- Uniquely number each of your requirements for easy identification and cross-referencing
- Highlight terms that are defined in Section 1.3 (**Definitions, Acronyms, and Abbreviations**) with **bold**, *italic* or underline
- For Deliverable 1, please highlight, in some fashion, all (you may have more than one) creative and innovative features. Your creative and innovative features will generally be described in Section 2.2 (**Product Functions**), but it will depend on the type of creative or innovative features you are including.

1 Introduction

This Software Requirements Specification (SRS) provides an overview of the software requirements for the Smart City Environmental Monitoring & Alert System (SCEMAS). SCEMAS is a software platform designed to collect, analyze, and present environmental data such as air quality, temperature, humidity, and noise levels to support city-level monitoring and decision-making.

This document outlines the system's purpose, scope, describes the characteristics of intended users, and the functional and non-functional requirements that guide the design and development of the system.

- Provide an overview of the document/SRS.

1.1 Purpose

- Specify the purpose of the SRS.
- Specify the intended audience for the SRS.

1.2 Scope

- Identify the software product(s) to be produced, and name each (e.g., Host DBMS, Report Generator, etc.)
- Explain what the software product(s) will do (and, if necessary, also state what they will not do).
- Describe the application of the software being specified, including relevant benefits, objectives, and goals.

  Features:

  - Reliable ingestion and processing of sensor telemetry via MQTT protocol, validating each incoming message for correct format, adherence to defined scheme and plausible value ranges.
  - All valid data is presistently stored in a database optimized for time-series data.
  - Preform real-time aggregation calculating meterics within geographical zones. Ex: 5min, 1h avg

1.3 Definitions, Acronyms, and Abbreviations

- Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS.
- This should be in alphabetical order.

1.4 References

X . (2026). "Project Outline for Software Design II(SE3A04) Smart City Environmental Monitoring & Alert System (SCEMAS)" [Online]. Available: https://avenue.cllmcmaster.ca/d2l/le/lessons/727030/topics/5250141 [Accessed: 14-Jan-2026]

- IEEE Style
- Provide a complete list of all documents referenced elsewhere in the SRS.
- Identify each document by title, report number (if applicable), date, and publishing organization.
- Specify the sources from which the references can be obtained.
- Order this list in some sensible manner (alphabetical by author, or something else that makes more sense).

1.5 Overview

Section 2 discusses the overall product description including the product functions, user characteristics, constraints, assumptions, and the perspective of the product relative to exisiting projects. Section 3 contains a use case diagram for the system. Section 4 highlights the functional requirements including the use cases organized by business event, different viewpoints for each event and associated scenarios, and a global scenario for each business event. Section 5 discusses the non-functional requirements including the look and feel requirements, usability and human requirements, performance requirements, operational requirements, maintainability, security, and political requirements.

- Describe what the remainder of the document/SRS contains.
  (e.g. "Section 2 discusses...Section 3...")

2 Overall Product Description

- This section should describe the general factors that affect the product and its requirements.
- It does not state specific requirements.
- It provides a *background* for those requirements and makes them easier to understand.

2.1 Product Perspective

- Put the product into perspective with other related products, i.e., context
- If the product is independent and totally self-contained, it should be stated here
- If the SRS defines a product that is a component of a larger system, then this subsection should relate the requirements of that larger system to the functionality of the software being developed. Identify interfaces between that larger system and the software to be developed.
- A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful

2.2 Product Functions

- Provide a *summary* of the major functions that the software will perform.
  - **Example**: An SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.
- Functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time
- Present the functions in a list format - each item should be one function, with a brief description of it
- Textual or graphical methods can be used to show the different functions and their relationships
  - Such a diagram is not intended to show a design of a product, but simply shows the logical relationships among variables

2.3 User Characteristics

- Describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise
- Since there will be many users, you may wish to divide into different user types or personas

2.4 Constraints

- Provide a general description of any constraints that will limit the developer's options

2.5 Assumptions and Dependencies

- List any assumptions you made in interpreting what the software being developed is aiming to achieve
- List any other assumptions you made that, if it fails to hold, could require you to change the requirements
  - **Example**: An assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

2.6 Apportioning of Requirements

- Identify requirements that may be delayed until future versions of the system

3 Use Case Diagram

- Provide the use case diagram for the system being developed.
- You do not need to provide the textual description of any of the use cases here (these will be specified under "Highlights of Functional Requirements").

4 Highlights of Functional Requirements

- Specify all use cases (or other scenarios triggered by other events), organized by Business Event.
- For each Business Event, show the scenario from every Viewpoint. You should have the same set of Viewpoints across all Business Events. If a Viewpoint doesn't participate, write N/A so we know you considered it still. You can choose how to present this - keep in mind it should be easy to follow.
- At the end, combine them all into a Global Scenario.
- Your focus should be on what the system needs to do, not how to do it. Specify it in enough detail that it clearly specifies what needs to be accomplished, but not so detailed that you start programming or making design decisions.
- Keep the length of each use case (Global Scenario) manageable. If it's getting too long, split into sub-cases.
- You are *not* specifying a complete and consistent set of functional requirements here. (i.e. you are providing them in the form of use cases/global scenarios, not a refined list). For the purpose of this project, you do not need to reduce them to a list; the global scenarios format is all you need.
- Red text below is just to highlight where you need to insert a scenario - don't actually write it all in red.

Main Business Events: List out all the main business events you are presenting. If you sub-divided into smaller ones, you don't need to include the smaller ones in this list.

Viewpoints: List out all the viewpoints you will be considering.

Interpretation: Specify any liberties you took in interpreting business events, if necessary.

BE1. Business Event Name #1

VP1. Viewpoint Name #1
Insert Scenario Here

VP2. Viewpoint Name #2
Insert Scenario Here

Global Scenario:
Insert Scenario Here

BE2. Business Event Name #2

VP1. Viewpoint Name #1
Insert Scenario Here

VP2. Viewpoint Name #2
Insert Scenario Here

Global Scenario:
Insert Scenario Here

5 Non-Functional Requirements

5.1 Look and Feel Requirements

5.1.1 Appearance Requirements

LF-A1. The application should use a distinct colour scheme that is in adherence with the companies brand colours.
Rationale: The application's UI should be consistent and professional to establish a sense of trust in the product's users [1].

LF-A2. All text in the application should be at least 14-px in size.
Rationale: The text in the application should follow font-size conventions to be large enough so it is easy to read for a wide range of individuals with varying vision levels [1].

LF-A3. The background colours of pages should contrast with the interactive elements (alert notifications, buttons, icons) directly on top of them.
Rationale: The interactive elements on the page should stand out from the background elements in order to indicate what can be interacted with on the page [1].

5.1.2 Style Requirements

LF-S1. The application's interface should be built for a resolution of 1920x1080 and be compatible with various screen sizes across desktop devices.
Rationale: As the interface will be used across many different desktop devices with varying screen sizes, the interface's size must be adaptable so all users can view the application as intended. 1920x1080 has been selected as the default size as it is the most common resolution on modern desktop devices [2].

LF-S2. The application's interface must utilize modern conventions of professional software UI/UX to achieve a professional and minimalist design.
Rationale: Adhering to commonly used UI/UX design principles will make the application look polished and professional, instilling trust in the user. Using well-known conventions of web design will also help the application to be more user-friendly [3].

LF-S3. All main pages of the application interface should be accessible from one menu (e.g. a navigation bar).
Rationale: The application should be simple to navigate, and users should not have to search through many menus to find crucial features [4].

5.2 Usability and Humanity Requirements

5.2.1 Ease of Use Requirements

UH-EOU1. The system must include a help menu for users, including a frequently asked questions page, and tutorial pages for how to navigate.
Rationale: A frequently asked questions page will be a helpful resource for users by providing answers to repetitive inquiries [5].

5.2.2 Personalization and Internationalization Requirements

UH-PI1. The system must allow users to configure personal preferences in the account settings. Features that must be selectable are light or dark mode and accessibility routes highlighted on the maps for users with disabilities.
Rationale: Enabling user-defined preferences improves the overall user experience and decreases task time [6]. The software will provide equitable utility for different users, making the application more inclusive.

5.2.3 Learning Requirements

UH-L1. The system must allow any third-party developer to successfully request and interpret environmental data within a two-hour time frame.
Rationale: A short learning curve ensures the API is intuitive and well-documented for the public. A short onboarding period encourages adoption of the REST API and improves developer productivity [4].

5.2.4 Understandability and Politeness Requirements

UH-UP1. The proficiency requirements for understanding any language being used in the system shall not exceed the B1 language proficiency level.
Rationale: The system application should be understandable to a wide range of people and a B1 proficiency level indicates the user can handle everyday situations and express opinions [7].

5.2.5 Accessibility Requirements

UH-A1. The system must have a setting to turn off the use of bright colours in the dashboard and map.
Rationale: Highly saturated colours are hard on the eyes and can cause visual fatigue when viewing for extended periods of time [8].

5.3 Performance Requirements

5.3.1 Speed and Latency Requirements

PR-SL1. The system shall ensure the latency of the data appearing on the City Operator dashboard does not exceed 1.5 seconds.
Rationale: Any delay in getting information to a city official directly increases the risk to public safety [4].

PR-SL2. The system shall authenticate a user and load their specific role-based dashboard in less than 3 seconds after the user clicks login.
Rationale: For City Operators, they would need to respond to events in a fast manner, as a delay in logging in during a crisis could delay critical decision-making [4].

PR-SL3. The Public API shall return the requested data within 1 second.
Rationale: Fast access to public data is essential for transparency and third-party developers. If the public-facing tools are slow, users are less likely to use the system for daily planning [9].

5.3.2 Safety-Critical Requirements

PR-SC1. N/A

5.3.3 Precision or Accuracy Requirements

PR-PA1. The system shall store and display all numerical data (e.g., temperature, humidity, and particulate matter) to at least two decimal places.
Rationale: High-resolution data is needed to identify subtle environmental trends. Rounding to the nearest whole number can hide small changes in events, which could prevent the alert of an event. This level of detail is necessary to detect the beginning trends before they cross the thresholds [4].

PR-PA2. The system shall represent sensor locations and alert markers on the geographical map with coordinate precision of at least 1 meter.
Rationale: As stated in Section 2.1, the system must allow officials to pinpoint exact locations of hazards. Low-precision coordinates could place an alert on the wrong area, leading to responders to the incorrect location [10].

5.3.4 Reliability and Availability Requirements

PR-RA1. The system shall maintain an availability of 99.99%, limiting the system updates and maintenance to no more than an hour a year.
Rationale: As the system is responsible for detecting hazards, it must be readily available. Despite the industry standard being an uptime of 99.99% [11], it is not attainable even for most services. Most cloud service providers' setups could achieve 99.99% availability [11].

PR-RA2. If the system or database experiences an unplanned failure, it shall automatically recover and restore core services within 120 seconds, without manual intervention.
Rationale: Disaster recovery aims to minimize business disruption by restoring service quickly after failures. This requirement sets an explicit RTO to reduce the operational "blind spot" during time-sensitive environmental incidents [12].

5.3.5 Robustness or Fault-Tolerance Requirements

PR-RFT1. The system shall store all incoming sensor data in a local cache for up to 48 hours during a network or internet failure. Upon reconnection, the system shall automatically upload this data to the main database in chronological order.
Rationale: Smart-city sensor networks must tolerate intermittent connectivity, including extended outages during major emergencies. IoT architectures commonly support offline operation by using local caching so telemetry is not lost and can be reliably forwarded when connectivity returns [13].

PR-RFT2. The system shall continue to provide core functions, even if non-critical subsystems become unavailable.
Rationale: If one part of the app breaks, the whole system should not crash. Using "graceful degradation" ensures that essential tools remain available [14].

5.3.6 Capacity Requirements

PR-C1. The system shall support at least 50 concurrent City Operators performing dashboard actions simultaneously without data collision, inconsistent state, or system failure.
Rationale: Multiple city operators may operate the dashboard simultaneously. The system must handle concurrent access safely to prevent conflicting updates (such as simultaneous alert edits) and ensure operational continuity under peak load [15].

5.3.7 Scalability or Extensibility Requirements

PR-SE1. The system shall be able to scale to support deployment in multiple cities of varying population sizes, with increases in the number of sensors, monitored zones, and concurrent users, without requiring an architectural redesign.
Rationale: Designing for scalability allows the system to grow in users, sensors, and coverage while maintaining performance and avoiding major redesign [4].

5.3.8 Longevity Requirements

PR-L1. The system shall be designed with a minimum operational lifespan of at least 10 years.
Rationale: This system is a large investment for a city, so to provide a high return on investment and ensure safety, the software must remain viable and maintainable without requiring a complete replacement [4].

5.4 Operational and Environmental Requirements

5.4.1 Expected Physical Environment

OE-EPE1. N/A

5.4.2 Requirements for Interfacing with Adjacent Systems

OE-IA1. The system must implement a read-only REST API endpoint allowing third-party and public access to non-sensitive aggregated environmental data.
Rationale: It is important for the system to be able to get important safety information out to the general public, and providing this API to other services will facilitate this communication [4].

OE-IA2. The system should be able to reliably receive and process data from designated sensors.
Rationale: The application must be able to receive sensor data in order to properly activate alerts and notify third party systems [4].

5.4.3 Productization Requirements

OE-P1. The application should be packaged in a zip file containing a README.
Rationale: The application should be quick to download and be easy to install and set up [4].

5.4.4 Release Requirements

OE-R1. The application should be compatible with the last 2 major versions of the Windows operating system.
Rationale: The system should be compatible with the most common desktop operating system worldwide [16].

5.5 Maintainability and Support Requirements

5.5.1 Maintenance Requirements

MS-M1. The system must be designed for modular component (microservices) upgrades independent of the entire system.
Rationale: A long maintenance window can result in missed alerts during critical environmental events. This will reduce risk and overall system downtime by completing individual software patches [17].

5.5.2 Supportability Requirements

MS-S1. The system must show application health metrics such as API latency, error rates, traffic and host saturation for all administrators.
Rationale: Administrators must be able to resolve data ingestion failures and offline issues immediately [18].

5.5.3 Adaptability Requirements

MS-A1. The system architecture must support the addition of new data ingestion for different environmental metric types.
Rationale: Environments are constantly changing and there will be new climate trends to be monitored. The system must be sustainable long-term and support easy additions without changing core logic [19].

5.6 Security Requirements

5.6.1 Access Requirements

SR-AC1. The system should only allow administrators to modify authenticated users and alert rules.
Rationale: There should be a way to add, manage, and delete users so that city operators can be successfully authenticated without public users having the same abilities. A Role-Based Access Control model [20] fits this need by defining a hierarchy of authentication levels through various roles.

SR-AC2. The system should only allow city operators and system administrators to view the dashboard.
Rationale: Public users should not be permitted to access all data through the system dashboard since this could reveal sensitive data to the public. A Role-Based Access Control model [20] fits this need by defining a hierarchy of authentication levels through various roles.

5.6.2 Integrity Requirements

SR-INT1. Data should be encrypted while in transit.
Rationale: Sensitive sensor data could be read if not encrypted leading to potential security risks and data leaks [21]. By keeping data encrypted, users and clients can trust the product to keep their data safe which will improve trust.

5.6.3 Privacy Requirements

SR-P1. The system shall not collect, process, or store any personally identifiable information (PII).
Rationale: This will ensure compliance with privacy regulations while reducing legal, ethical, and security risks [21].

SR-P2. The public API shall only expose non-sensitive environmental data aggregated by city zone.
Rationale: Revealing precise sensor data to the public could reveal the location of sensors used in the city [21]. Giving aggregated data balances transparency with the security of the system.

5.6.4 Audit Requirements

SR-AU1. The system shall maintain an immutable log of all significant events, including alert triggers, alert acknowledgement and alert rule modifications.
Rationale: Maintaining an unmodifiable log of all events can be used to audit the system and to track administrative changes. This follows the guidelines for cybersecurity set out by the Government of Canada [21].

5.6.5 Immunity Requirements

SR-IM1. Public-facing interfaces shall enforce rate limits to protect against denial-of-service attacks and excessive data scraping.
Rationale: Limiting the rate public users can access the API should reduce the strain on servers and limit the risk of a successful denial-of-service attack. This method of resisting denial-of-service attacks is outlined in ITSG-33 (Security Control SC-5) [21].

5.7 Cultural and Political Requirements

5.7.1 Cultural Requirements

CP-C1. The system shall support all official languages of the country it is operating in.
Rationale: National language laws state that for government systems all official languages must be available [22].

CP-C2. The app and system will not use any offensive or hurtful language towards any individual or people group based on their; citizenship, gender, sexual orientation, religion, political views, disability status, or any other common identifier.
Rationale: Hate speech laws must be followed and all users should feel safe when using the app [23].

5.7.2 Political Requirements

CP-P1. The public facing app should not use fear inducing terms when describing alerts. Only scientifically backed terms will be used.
Rationale: To not create panic or fear when a fire or pollution alert is out [24].

5.8 Legal Requirements

5.8.1 Compliance Requirements

LR-COMP1. The app will not process any raw audio data. IoT devices will process audio and never transmit raw audio streams only sound levels.
Rationale: Live audio streaming without consent is wiretapping under the Personal Information Protection and Electronic Documents Act and the Criminal Code of Canada Section 184 [25] [26].

LR-COMP2. The app's public facing app will meet the AODA accessibility requirements.
Rationale: The app must meet the accessibility requirements by law [27].

LR-COMP3. All app data will be stored on servers inside the country of origin.
Rationale: This is the law on how municipalities have to collect and store data [27].

LR-COMP4. The app will contain immutable audit logs and have frequent backups all while being secure for a minimum of 7 years.
Rationale: The law states municipalities have to retain records in a secure manner to fulfill freedom of information requests and audits [28].

LR-COMP5. Environmental action taken by a user in the dashboard when responding to an event has to be legal.
Rationale: All environmental actions regarding pollution, toxicity, and waste, need to follow the Canadian Environmental Protection Act [29].

5.8.2 Standards Requirements

LR-STD1. The app will follow the W3C WCAG 2.1 Accessibility Guidelines.
Rationale: This is to meet AODA web standards [30].

6 Innovative Feature

The chosen innovative feature we will be implementing is a public interface for our SCEMAS application that will allow any public user to view dashboards which will visualize all of the publicly available environmental metrics available from the REST API. This feature will benefit the product by increasing the number of users for our application. Since the REST API will be mainly used by third-party developers, we want to allow regular public users to also view and be informed of environmental issues. We have also come up with other innovative features, as listed below:

- Provide access to external resources that guide users on how to navigate specific environmental concerns whenever a warning is issued.
- The dashboard map uses highlighted areas to emphasize zones that require extra precaution, such as school zones and residential neighbourhoods.
- An integrated RAG (Retrieval-Augmented Generation) model that leverages city documentation, such as covering water treatment, waste management, snow removal, and emergency services, to provide intelligent suggestions. Users can review these insights before choosing to take action via dedicated buttons, such as calling 911 or routing the issue to the appropriate department.
- The system generates insights based on real-time weather data. For example, it can identify heavy snow accumulation and automatically recalculate and transmit optimized routes to snow plow operators.
- Administrators can utilize biometric authentication, including fingerprint and retinal scans.
- Rather than simply displaying current sensor data, the platform predicts potential risks by analyzing trends. If particulate matter concentration begins to rise rapidly toward the 100 $\mu g/m^3$ threshold, the system will preemptively alert operators before the limit is officially breached.
- Notification settings allow organizations to assign specific employees to monitor particular alert types, such as air quality, or to oversee designated geographic regions.
- The interface fully supports both light and dark modes to accommodate different lighting environments and user preferences.

A Division of Labour

This sheet indicates the contributions of each team member and is signed by all team members.

Khan, Saqib

- Introduction
- System Diagram
- Use-case Diagram as a group
- 2.4 Constraints
- Brainstorming requirements, business events
- BE2. Modify Existing Alert Rule
- 5 Non-Functional requirements discussion
- 5.1 Look and Feel Requirements contribution
- 5.3 Performance Requirements
- Discussed Innovative feature

Abdullah, Suzanne

- 2.3 User Characteristics
- System Diagram contribution
- Use-case Diagram with group
- BE6. Checking Audit Log
- 5.1 Look and Feel Requirements
- 5.4 Operational and Environmental Requirements
- Innovative feature main idea
- Brainstorming requirements, business events

Olejniczak, David

- 1.4 References + formatting references in Non-Functional Requirements
- 2.1 Product Perspective Paragraph
- Use-case Diagram as a group
- 2.4 Assumptions
- BE1. Creating an Alert Rule
- BE5. Public API Request to Access Data
- BE7. User authentication
- 5.7 Cultural and Political Requirements
- 5.8 Legal Requirements
- Brainstormed Innovative feature with group

Lai, Vanessa

- 1.2 Scope
- BE4. View Dashboard
- 5.2 Usability and Humanity Requirements
- 5.5 Maintainability and Support Requirements
- State Diagram

- Use-case Diagram as a group
- Brainstormed Innovative feature with group
- 6. Innovative Feature

Buehlmann, Lukas

- 1.5 Overview
- 2.2 Product Functions
- Use-case Diagram as a group
- BE3. Sunlight is very high on the UV index
- 5.6 Security Requirements
- Brainstormed Innovative Ideas with group
- Reviewed and edited final document