Preliminary Software Requirements Specification

for

Vigilance – A public space monitoring system

Version 1.0

Course: Software Architecture

University of Oulu

Table of Contents

1. Introduction

This is a Preliminary Software Requirements Specification (SRS) document for *Vigilance* – A public space monitoring system. As opposed to a formal SRS document, this document provides the basic understanding and requirements of the system to be designed. A full-fledged list of requirements classified as per their types will not be part of this document. This document contains subsections for the overall product description, hardware and software requirements, as well as list of references used, if applicable. Additionally, product functions, user characteristics, software and hardware constraints are among the topics that are discussed in this document.

1.1 Purpose

This document's purpose is to state the required functionality of *Vigilance* – A public space monitoring system, in terms of its inputs and outputs, as well as to devise criteria to fit the customer's requirements. The intended audience for this document consists of engineers responsible for the design and development of this system, as well as customers and relevant stakeholders of this project.

1.2 Product Scope

The *Vigilance* system will be deployed at a subway station, as part of the larger smart-city drive undertaken by the municipality of the Raccoon City. Leveraging a cloud of sensors embedded throughout the Raccoon City subway station, *Vigilance* should aggregate the raw data and analyze them towards a higher level of abstraction. For example, smoke detectors can only alert users of presence of smoke. However, when combined with visuals of fire, provided by cameras, *Vigilance* can aggregate and analyze them to estimate an incident of higher abstraction called "Fire". Similarly, the system should leverage the wide array of sensors to inform the authorities of unwanted presence, especially after an area has been evacuated. Speaking of which, the system should also send alerts when it detects a hazardous incident, and/or when it has to notify people at the subway station of an immediate evacuation. Evacuation may be triggered by the incident analyzed by the system, or the operators of the system may manually trigger it. For this purpose, the *Vigilance* system should interface with a system called *Core*, which is already implemented and managed by the municipal authorities. The *Vigilance* system only needs to inform the *Core* system of the incidents it has processed and analyzed, along with the estimations of its severity and probability. In addition, the *Vigilance* system will also inform about the levels of alerts that need to be broadcast to the relevant authorities (police, fire department, etc.), and to the general public who may have subscribed to incident alerts. These alerts are managed by the *Core* system only, and *Vigilance* only needs to send the alert requests to it. To pilot the *Vigilance* system, the authorities want to utilize the cloud of sensors to detect incidents like fire, and be able to assess the situation to provide its best estimate of these incidents and the need for alerts.

1.3 Definitions, Acronyms, and Abbreviations

Vigilance – The system to be developed. It leverages the cloud of sensors that are deployed throughout the subway station, to detect hazardous incidents and undesired presence. The system also informs the authorities of these incidents, along with alerts based on the varying levels of severity of the incidents the system has determined.

Core – This is the system that has already been implemented, and is operational in the city. It's a city-wide system that interfaces with different smart systems and processes their requests, like informing authorities and broadcasting alerts.

Closed-Circuit Television (CCTV) – It is the use of video cameras for video surveillance.

4G – Fourth generation of wireless mobile telecommunications. The protocol that *Vigilance* will use to communicate with the sensors, and what *Core* will use to send out alerts to general public.

Graphical User Interface (GUI) – A form of user interface that allows the user to interact with a system using icons and other related visuals.

Application Programming Interface – A software intermediary that allows two applications to communicate with one another.

“Not only SQL” (NoSQL) - These are non-tabular databases that store data differently than relational databases like SQL.

1.4 Organization

The remaining part of this document provides further detail about the product, overall requirements, and its potential users. In section 2, we will provide the context and functions of the product, followed by any potential hardware and software constraints and any assumptions concerning the operational environment. In section 3, we will list the different interface requirements for the system.

2. Overall Description

This section will give a detailed breakdown of how the *Vigilance* system will exist in the context of its operating environment. This section will also cover the functionality of this product, necessary and adequate enough for the architecture design. User characteristics and expectations will also be defined, in addition to various constraints dictating *Vigilance's* functions and operational characteristics, and assumptions and dependencies.

2.1 Product Perspective

In an effort to enhance Raccoon city's safety and security, in line with its drive to deploy intelligent agents to accomplish these tasks, the *Vigilance* system will help the city monitor and manage emergency response system for its Raccoon subway station. Relying on the sensors spread throughout the station, the system will monitor the subway space and trigger emergency alerts based on the analysis of different raw signals generated by individual sensors. The system will leverage sensors that measure raw signals like temperature, humidity, smoke, noise, and CCTV cameras for visuals. With the help of machine learning techniques, the *Vigilance* system will aggregate and analyze these raw signals to produce the probability of an abstract incident (e.g. fire, unexpected presence), based on which it will assess the level of emergency response and alerts needed. The system will have its operators, alongside the operators of the *Core* system, so they can intervene in the event of false alerts. These operators can also override the system altogether, if they see the system has failed to detect and /or analyze an incident. The operators will trigger the alerts based on the level of emergency response needed.

2.2 Product Functions

Following are the major functions needed to operationalize the *Vigilance* system:

2.2.1 Raw signal processing and analysis

Using the raw signals from individual sensors like CCTV cameras, infrared sensors, smoke detectors, etc., the *Vigilance* system should aggregate and analyze them. The system must use in-built algorithms that utilize signal processing and audio and image processing to produce a composite incident with a probability measure, and the corresponding severity of alerts and emergency response needed. In case an evacuation order is recommended, the system should activate presence dictation in the area that has been evacuated.

2.2.2 Alerts

The *Vigilance* system interfaces with the city's *Core* system, sending the composite incident and the accompanying assessment of the situation that will trigger an emergency response. Unless interrupted by the operators of the systems, the *Vigilance* system will send command to the *Core* system to trigger an alert. In case of an evacuation, the system alerts the operators, the authorities, and the general public. The public receives the alerts as an SMS, which is also routed through the *Core* system, as it has the necessary provisions and security clearance to broadcast alerts to general public in the event of an emergency.

2.3 User Classes

Users of the *Vigilance* system are the operators who have the necessary skill and the knowledge of the underlying systems. As the system is used to ensure public safety and security, it is critical that its users are aware of the system's nuances. The general public will never come in contact with the system, and only be a recipient of the alerts that the system triggers.

2.4 Constraints

The system will be deployed within the wireless network that currently hosts the city's *Core* system. Therefore, the *Vigilance* system will share the same hardware, software, and network constraints that are applicable for the *Core* system. *Core* is a java-built system that runs on Windows platform only, uses a NoSQL database, and relies on the city-wide 4G network for communication.

Although *Vigilance* will be getting real-time data stream from a wide array of sensors, secondary storage may not be critical. However, for future training of the machine learning algorithms employed in the *Vigilance* system, it is necessary that every composite incident analyzed by the system is stored for future reference, including the context details like source sensors sending the raw signals, timestamps, etc. In this regard, adequate secondary storage space can be added alongside *Core* storage infrastructure. However, the most critical element of the *Vigilance* system is the main memory, which should, at minimum, be 32 GB to be able to handle processing of data from the sensors and cameras.

2.5 Assumptions and Dependencies

The *Vigilance* system depends heavily on the sensors deployed across the Raccoon subway station. The proper functioning of these sensors is critical for the system to do its job. Furthermore, the machine learning techniques employed to process signals and build a composite incident have been comprehensively tested and for their accuracy. As highlighted earlier, *Vigilance* will interface with the *Core* system to trigger alerts. Any issues with the interfacing may cause delay in responding to emergency, or even miss them completely.

3. External Interface Requirements

3.1 User Interfaces

Vigilance should have an intuitive yet comprehensive GUI for its operators to use. The GUI should enable its users, i.e., the operators, respond to emergencies with as few clicks as possible. The events processed and analyzed by the system, and the accompanying values of probabilities and the potential severity of the incident, should be emphasized, and the immediate action recommended by the system should be presented as clearly as possible. The user interface will be displayed on the Windows-based workstation manned by the operators.

3.2 Hardware Interfaces

The *Vigilance* system will interface with the cloud of sensors deployed at the subway station. The resource-intensive processing of sensor data and the subsequent analysis will be done on the server side, while the user interface on the workstation will only act as a receiver of the analysis and display the same in sufficient detail to the operators.

3.3 Software Interfaces

Vigilance will interface with the *Core* system for alerts-related processing. A simple API can be used to establish this interface. In addition, the system hosting *Vigilance* will run on Windows Server, and use a NoSQL database for storage purposes.

3.4 Communications Interfaces

Vigilance will use secure wireless protocols to communicate with the sensors. In case of interacting with the *Core* system, the communication may happen over a wired or wireless network, as both the systems are very likely to be housed in the same facility.