SYSC3010 Computer Systems Development Project
FANS Project Proposal

Group L1-G8

[1]

Grant Achuzia, 101222695
Javeria Sohail, 101197163
Matteo Golin, 101220709
Saja Fawagreh, 101217326
TA: Sean Kirkby

Created: February 11th, 2024
Modified: February 12, 2024

Contents

1 Introduction

This document presents the proposal for Fire Alarm Notification System (FANS), a comprehensive solution addressing the alarming rate of preventable fire deaths in Canada. [2] The L1-G8 group has developed FANS as a multi-device fire alarm system with advanced features to significantly reduce fire deaths through timely detection and response to fire emergencies.

The proposal reveals the detailed system design and comprehensive work plan for the development and implementation of FANS to reduce the number of preventable fire-related deaths in Canada. It outlines the system's architecture and provides a structured work plan detailing the phases of development, testing and deployment to ensure the timely release of FANS to effectively mitigate fire disasters.

1.1 Background

Every year in Canada, 220 people die from fire-related causes. [3] At least 1 in 7 of these deaths occurred in homes without a working smoke alarm. [3] This is an alarming number of deaths, which FANS aims to prevent.

As a result of these preventable deaths, the government of Canada has released guidelines for smoke alarm installation and care. [2] FANS aims to take a modern approach to fire safety by integrating its smoke alarm system with the Internet of Things (IoT).

Traditional fire alarm systems often lack modern communication features and real-time monitoring capabilities. These limitations contribute to delays in response time and can lead to preventable fire-related deaths. Modern systems make use of not only smoke detection sensors, but also heat detectors that can detect a rise in temperature due to fires. [4] Smoke detection systems are also subject to unsafe disarming by users who have their alarm installed close to their kitchen space. False alarms can be frustrating while cooking, but disabling the smoke alarm is a dangerous action that puts the user at risk of forgetting to re-enable it. [5]

The FANS project aims to address the shortcomings of traditional systems and implement the strengths of modern systems by introducing a comprehensive multi-system solution. FANS will use both smoke and temperature sensors to cover situations where smoke cannot reach the detecting device. It will issue real-time notifications via SMS and email to notify homeowners of an emergency. Finally, FANS will offer configurable thresholds for smoke and temperature alarms, as well as configurable timeouts for the system to prevent disgruntled users from unsafely disabling their alarm system while cooking.

By leveraging technological advancements, FANS seeks to improve the effectiveness of fire alarm systems and reduce fire-related fatalities in Canada.

1.2 Motivation

The motivation for the FANS project is to address the critical problem of preventable fire-related deaths in Canada. Fire safety is a fundamental concern for individuals, families, and communities nationwide. By leveraging modern technological advancements, the FANS system aims to significantly enhance the effectiveness of fire alarm systems, reducing response times, and ultimately saving lives. The importance of this project cannot be overstated, as it directly impacts the safety and well-being of Canadians.

1.3 Project Objective

The overarching objective of the FANS project is to develop and deploy a functional fire detection and notification system that enhances fire safety. FANS accomplishes this by providing real-time fire detection, timely notifications to users, and proactive monitoring of environmental conditions.

1.4 Specific Goals

The goals that the FANS development team aims to achieve in this project are as follows:

1. Develop and test a smoke detection algorithm on a Raspberry Pi to ensure accurate and reliable smoke detection.

2. Integrate smoke and temperature sensors on the Raspberry Pi to a real-time database for real-time data collection.

3. Implement a robust notification system capable of triggering alarms at installation sites and sending email alerts to users in affected areas.

4. Allow configurable smoke and temperature alarm thresholds for more control over alarm sensitivity, as well as configurable time-out durations on detection units that are close to cooking spaces.

5. Create a user-friendly web interface for system monitoring and interaction, allowing users to access real-time data visualizations on their dashboard. Users should be able to configure system settings, add new user contact information to the notification subscriber list, and receive alerts all in the native web UI.

1.4.1 Functional Requirements

1. Consistent accuracy from the smoke sensors to avoid any false alarms/triggers.

2. Collected sensor data must be written to the cloud database in real-time for accurate monitoring.

3. Email and SMS notifications must be sent to all users in the notification subscriber list.

4. Users will be able to configure smoke and temperature alarm thresholds from the web interface.

5. Users will be able to set variable length time-outs for the detection unit while they cook. The detection unit will become active immediately after the time-out ends.

6. The web interface must provide time series charts that display both smoke levels and temperature levels as data is written to the real-time database.

7. The detection unit must be able to signal an emergency to the both the alarm unit and the notification unit over the local network.

1.4.2 Non-Functional Requirements

1. The system must be highly reliable and available to ensure immediate response during emergencies.

2. The system should scale easily to accommodate a growing number of users and installation sites.

3. There must be data security and privacy measures to protect sensitive contact information.

2 System Design

2.1 System Overview Diagram

Figure 1 shows the deployment diagram for FANS. There will be three devices which communicate with each other on a local network:

1. The alarm system on a Raspberry Pi 4.

2. The smoke detection system on a Raspberry Pi 4.

3. A notification system on a Raspberry Pi 4.
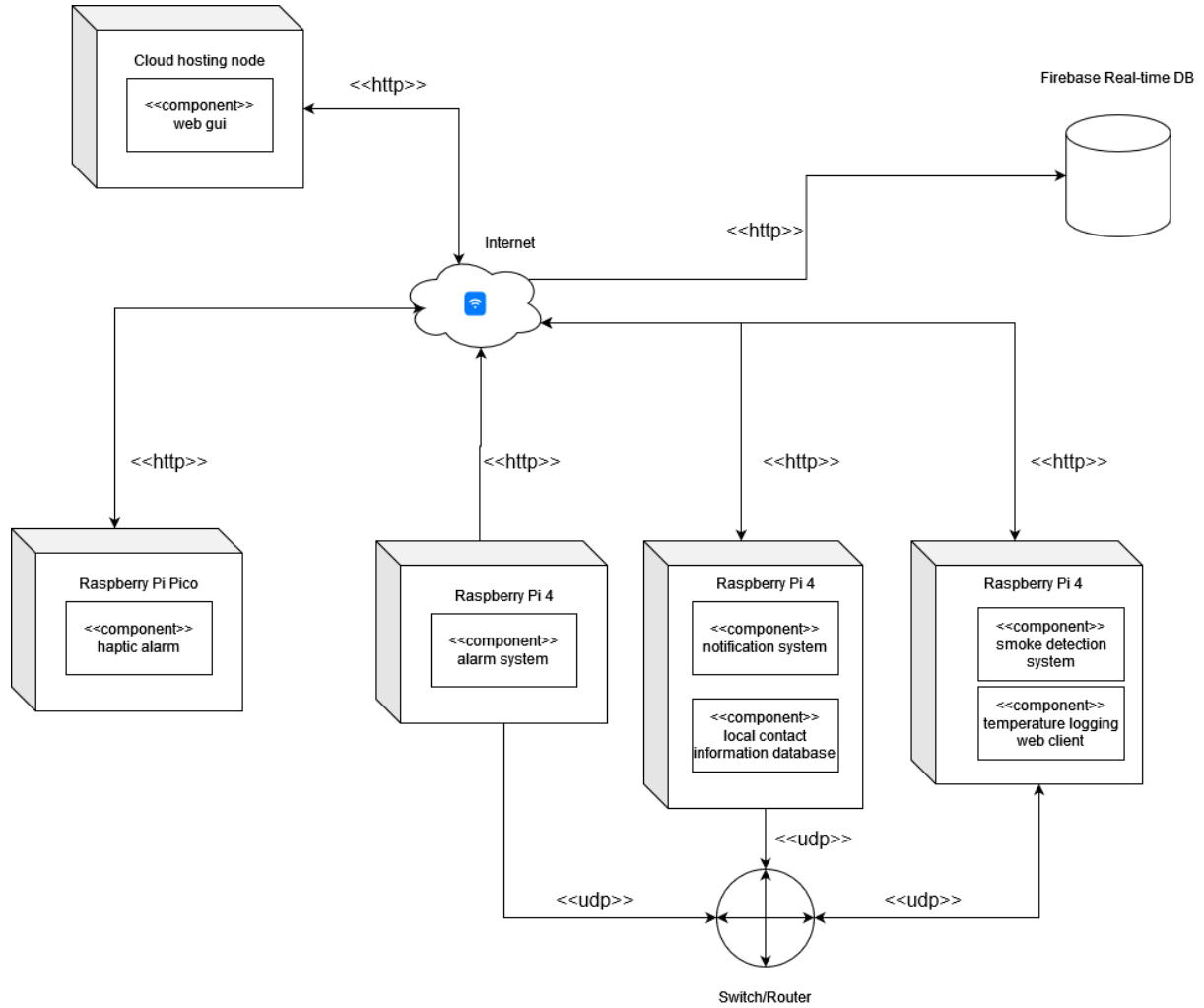
Figure 1: Deployment diagram of FANS.

The smoke detection and temperature logging system will continuously update the cloud database with recorded smoke and temperature levels.

When the smoke detection system detects a smoke or temperature measurement above the limiting threshold, it will signal the alarm system to sound the alarm using a UDP packet. It will also signal

the notification system to send out notifications to the subscriber list in the same way, and then send a write request to the cloud database to signal the emergency detection flag.

The fourth device, the Pi Pico, will buzz when the cloud database indicates that an emergency has been detected by the detection system. This will physically notify the user of the emergency.

The notification and alarm system are on separate devices to preserve their integrity in the case of an emergency. It is possible that the smoke detection system will be closer to the source of the fire, and therefore be at risk of being damaged/disabled. With the notification and alarm systems on separate devices connected through a local internet connection, they will be able to perform their responsibilities with less risk of being damaged by flames. The local connection also allows the alarm to sound in the case where the local network loses internet connection.

The cloud database is a real-time database to facilitate the real-time nature of display information on the web GUI. In addition, it will allow the haptic feedback device to buzz as soon as the smoke detection system signals an emergency to the cloud database. The fire alarm system has a real-time nature which needs to be preserved.

The web GUI will be served on the cloud, allowing us to serve our interface to multiple users more easily.

Scalability
This configuration would allow us to scale our system horizontally very easily. Multiple different alarm systems and smoke detectors could be connected over a local network to a notification system, allowing smoke detection in multiple rooms and alarms sounding in multiple rooms. Additionally, several haptic alarm wearable devices could be clients to the real-time database, allowing any number of users to be notified haptically in the event of an emergency.

Modularity
Each subsystem has a single responsibility that allows for modularity in FANS. Subsystems have a clearly defined interface with specific messages passed between them, allowing subsystems to operate without knowledge of each others' implementations.

The smoke detection system simply sends emergency signals out over the local network to nodes who are listening, which allows it to set off multiple alarm nodes.

The cloud database makes data available in such a way that multiple web UI clients and multiple wearable devices can access sensor data and emergency flags.

The notification system is entirely self-contained, and only requires access to contact information in the cloud database.

2.1.1   Communication Protocols

The smoke detection system will be communicating with an array of temperature and smoke sensors using the I2C protocol over the GPIO pins of a Raspberry Pi 4.

Each node on the local network (the smoke detection system, alarm system and notification system) will communicate with each other using UDP packets over the local network.

Nodes which communicate with the cloud hosted real-time Firebase database will use HTTP requests to interact with the database.

- The smoke detection system will post JSON payloads to the database to write data.

- The web GUI will request JSON over HTTP to update its visual components (dashboard with charts, etc.) with the data stored in the database.

- The notification system will request contact information from the database over HTTP to send notifications to affected users.

- The haptic alarm system will poll a database flag over HTTP to check if there is an active emergency.

Finally, the notification system will communicate with affected users over email and SMS text notifications. This will use standard internet, SMS and email protocols.

2.2   Component Details

The following section will describe the interfaces and responsibilities of each major node in the FANS system.

2.2.1   Smoke Detection System

The first component in the proposed system is the smoke detection and temperature sensing system. It will read sensor data from a locally connected smoke sensor and temperature sensor using the I2C protocol over the Raspberry Pi 4's GPIO pins. It will then update the real-time Firebase database with the collected sensor data over an internet connection.

If any temperature or smoke measurements are above the critical threshold for signalling an emergency, it will send a message to the notification system and the alarm system over UDP to signify an emergency. It will also update a flag in the real-time database to signify that an emergency has been detected.

The smoke detection system can be configured by the user. The user configuration settings will be read from the cloud database whenever they are updated, which allows the smoke detection system to use configurable smoke and temperature thresholds and have a time-out.
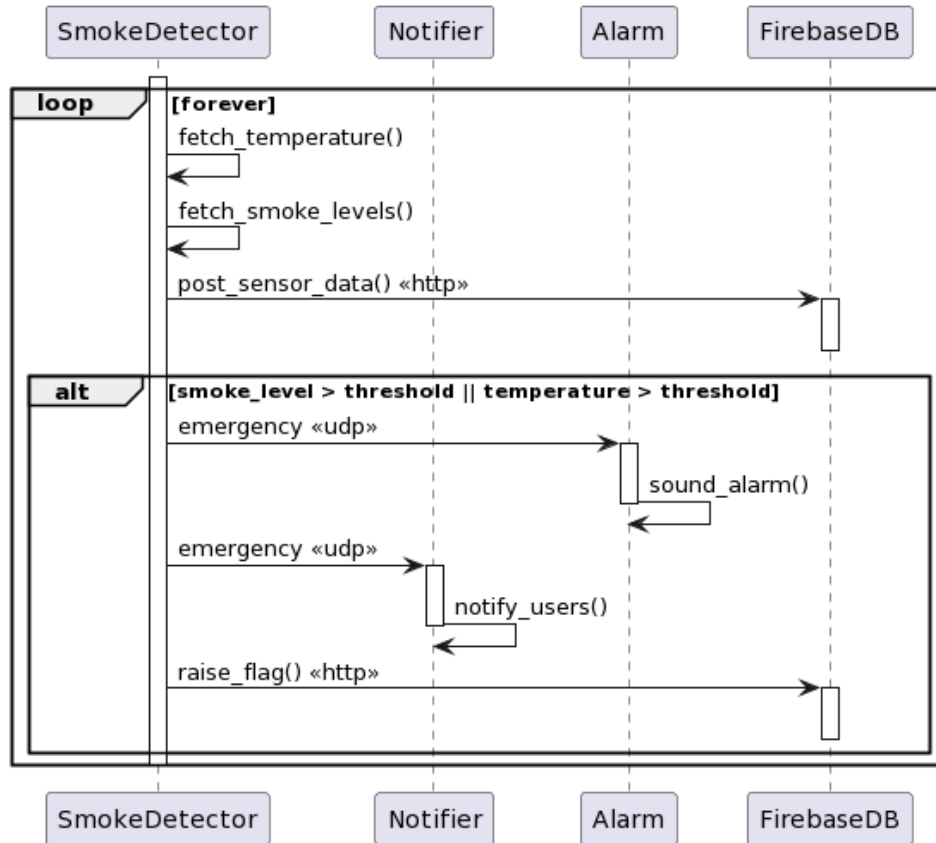
Figure 2: Sequence diagram for the primary responsibilities of the smoke detector system.

2.2.2 Notification System

The notification system is responsible for notifying affected users of an emergency and triggering the alarm system. It will also send SMS and email notifications to all users in the database to notify them of the detected emergency.

The notification system sends its notifications following the receipt of a UDP message from the smoke detection system. The contact information of the notification recipients are stored both on its local database and the cloud database. The local database is synchronized periodically with the cloud.

2.2.3 Alarm System

The alarm system is responsible for sounding an alarm in the building when an emergency has been detected. The alarm system receives notification over UDP from the smoke detection system when an emergency has been detected, at which point it sounds an alarm and flashes lights.

The lights will be an on-board SenseHat, and the alarm is an audio-hat module. The alarm will continue to sound until the system receives a UDP message to stop.

2.2.4 Haptic Alarm System

The haptic alarm system is designed to be a wearable device that vibrates during an emergency. The device queries the real-time database for a flag indicating an alarm. Once the flag is raised, the device vibrates until the emergency has ended (the flag is "lowered" in the database).
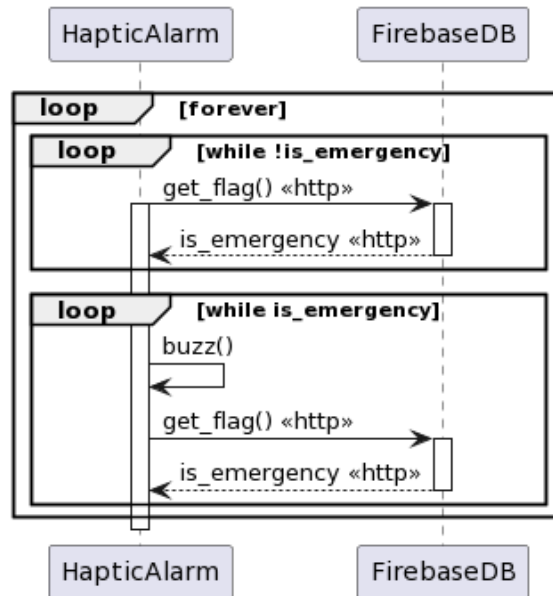

Figure 3: Sequence diagram for the primary responsibilities of the haptic alarm system.

2.2.5 Firebase Real-Time Database

The real-time database is responsible for storing smoke and temperature level data in real-time, as well as a database flag indicating emergency. User contact information will also be stored in this database. Finally, it will also contain user-configured settings (such as smoke level threshold for triggering an emergency). It will be cloud hosted by Firebase.

The interaction between the haptic alarm system and the real-time database is visible in Figure 3. The interaction between the smoke detector system and the real-time database is visible in Figure 2. The primary interaction between the web GUI and the real-time database is documented in Figure 4, and secondary interactions can be seen in Section 2.3.

2.2.6 Web GUI

The web GUI is responsible for displaying the sensor information from the real-time database on time-series charts. It will also allow users to configure settings for FANS (emergency thresholds, etc.) and add more user contact information to the notification subscriber list. It will also allow users to disable the alarm when an emergency has been resolved.
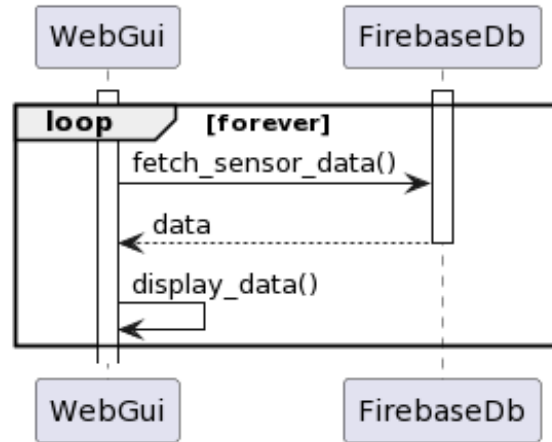
Figure 4: Sequence diagram for the primary responsibilities of web GUI.

2.3 Use Cases

2.3.1 Trigger Emergency

Flow of events

1. The smoke detection system sends a TCP message to the notification system to warn of an emergency.

2. The smoke detection system raises a flag in the real-time database to indicate the emergency.

3. The notification system forwards this message to the alarm system.

4. The notification system begins sending SMS and email notifications to the users who are listed in the database.

5. The alarm system receives the TCP message from the notification system.

6. The alarm system sounds the alarm and flashes LEDs.

7. The haptic alarm system reads the flag from the real-time database.

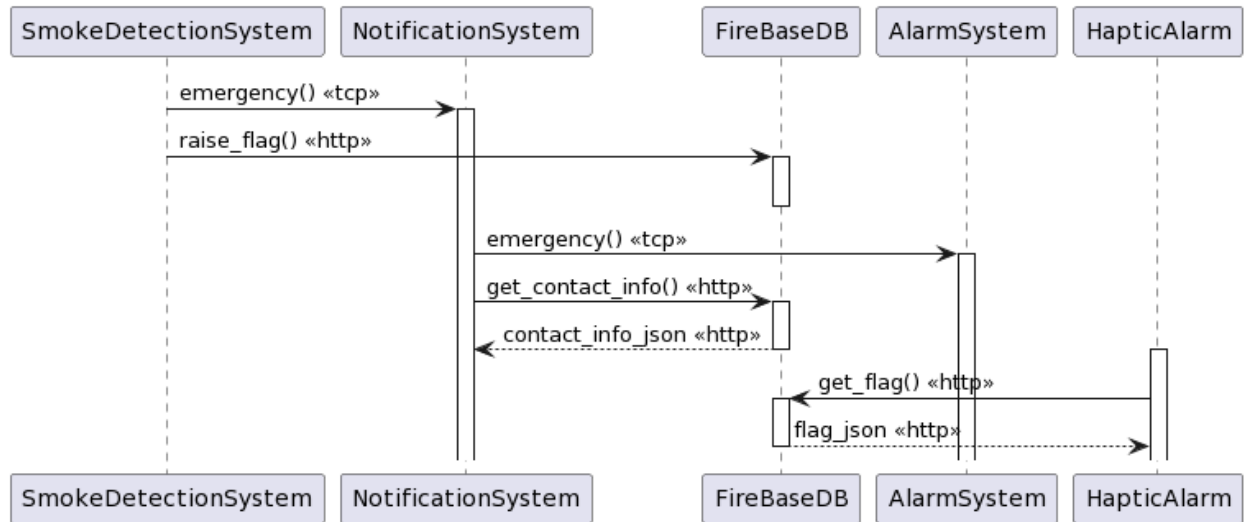8. The haptic alarm system begins buzzing.

Figure 5: Sequence diagram for the major interactions in the "Trigger Emergency" use case.

3 Work Plan

The following section describes the approach taken by the FANS development team to collaborate effectively and complete the FANS system before its deadline.

3.1 The Project Team

Each member of the project team is a third year computer systems engineering student.

Grant Achuzia
Grant's strengths lie in effective project management and organization, and he is passionate about delivering successful projects in academic settings. Grant also has significant experience designing web interfaces that are accessible and aesthetically pleasing.

Matteo Golin
Matteo has background in developing embedded system on Raspberry Pi's from his work at Carleton University's rocketry design team, where he is leading the development of a real-time telemetry system using Blackberry's QNX RTOS.

Saja Fawagreh
Saja has developed skills in user interface design from her time on co-op, and excels and making visually appealing interfaces because of her sharp attention to detail.

Javeria Sohail
Javeria has expertise in system architecture design, and has a strong knowledge of object-oriented design patterns. She is passionate about optimization, efficient design and modularity.

3.1.1 Roles and Tasks

Team Member	Major Role/Task (Primary)	Secondary Responsibility (Testing)	Reason for assignment
Grant Achuzia	User interface design	Local network communication	Very familiar with HTML/CSS/JavaScript and has knowledge of good web design practices.
Matteo Golin	Sensor data collection	Local network communication	Experience writing software to drive sensors and collect data from them because of his experience with CU InSpace. Familiar with networking protocols from 5G automation co-op at DELL Technologies.
Saja Fawagreh	Notification system	Local network communication	Good communicator, capable of writing succinct Python logic for accomplishing the task.
Javeria Sohail	Haptic alarm system	Local network communication	Familiar with LED and buzzer actuators from SYSC3310.

Table 1: Assignment of team members to major project tasks

3.1.2 Teamwork Strategy

Regular Team Meetings
We will schedule regular team meetings to discuss project progress, share updates, and address any challenges or questions. Our team will establish MS Teams as our main communication channel to ensure efficient communication within the team and with the TA.

Additionally, the team meetings will be documented to review what we talked about in the case that someone misses a meeting.

Version Control
We will use GitHub to manage our codebase changes. This will include practices such as creating branches for features or bug fixes, committing regularly, and requiring code reviews before merging pull requests.

Informative commit messages that follow the SYSC3010 commit formatting guidelines will be required to avoid misunderstandings.

The team will release frequent prototypes using GitHub's version release features. Versioning will follow *semantic versioning* (semver) notation. [6]

Code Reviews
Before merging code changes, thorough code reviews will be conducted to ensure code quality and maintainability, as well as gain feedback about feature implementations from other team members.

Testing
We will follow a testing strategy that includes unit testing, integration testing, and system testing to ensure the reliability and robustness of the FANS.

FANS will be frequently prototyped to ensure a working system at each development step. This methodology is in accordance with the V-model. [7] To ensure this, mocking will be used to simulate modules of the system that haven't been fully implemented.

Unit testing will be written by developers who have written the feature under test, and will also be run as GitHub workflows on pull requests to ensure that new code does not break existing functionality.

Feedback
Team members will provide constructive feedback to each other for continuous improvement and learning (also making use of Feedback Fruits). We will also request feedback from our TA at every milestone so that our final system comes together seamlessly.

3.1.3   What We Will Need to Learn

Hardware integration
Since the FANS includes both hardware and software components, we need to learn how to integrate them seamlessly. We utilize online resources and forums where hardware integration techniques are discussed. Many of our hardware modules (SenseHat, AudioHat, etc.) have associated online tutorials and a plethora of information associated with them. [8] We will leverage this information.

Real-time communication
Implementing real-time communication features such as SMS and email notifications requires knowledge of APIs and protocols for sending messages. We will look at the documentation and instructions provided by communication service providers (e.g. SMS) and learn how to integrate these services into our system.

We will also seek out Python tutorials for automated email sending, as there are several Python libraries that provide this functionality (such as `smtplib` [9]).

Developing a web interface
Creating a user-friendly web interface for the FANS requires knowledge of front-end development, including HTML, CSS and JavaScript. We will use online tutorials, courses and documentation of web development frameworks such as React.

Test strategies
Developing effective testing strategies, including unit testing, integration testing, and system testing, requires an understanding of various testing frameworks and methodologies. We will explore tutorials for language specific testing frameworks, and use online resources to teach ourselves how to effectively mock components of our system.

3.2 Project Milestones

Milestone Name	Date	Description
UI	2024-02-20	Develop a user-friendly web interface for monitoring system status, configuring thresholds, and interacting with the system.
Database	2024-02-23	Set up a database to store employee contact information and system data, ensuring efficient management and retrieval.
Smoke detection	2024-03-05	Develop and test the smoke detection algorithm on the Raspberry Pi to ensure accurate detection of fire and smoke.
Sensor integration	2024-03-09	Integrate sensors with the Raspberry Pi for real-time data collection, enabling the system to monitor environmental conditions.
Notifications	2024-03-12	Implement a notification system to trigger alarms and alert users via SMS and email in case of fire emergencies.
Testing	2024-03-20	Conduct comprehensive testing to ensure the functionality and integration of both hardware and software components.
Finalization	2024-03-27	Finalize system configuration, including thresholds and notification settings, and conduct comprehensive testing.

Table 2: Major milestones for FANS development.

3.3 Schedule of Activities

Under construction!

4   Project Requirements Checklist

Computer Components

- 1 Raspberry Pi per each of the following: Smoke Detector device, Smoke Alarm, Wearable Accessory

- 1 Raspberry Pi will be used to transfer info to and from all the other RPis to the Web GUI (running in headless mode). There are 4 Raspberry Pi's used in the project overall.

Hardware Components

- MQ2 Smoke Sensor (x1), LCD Display (x2), AudioHat Module (x1), SenseHat Module (x1)

- Actuator: LCD Display, AudioHat Module

- Smoke sensor waits till it senses smoke, once that requirement is met it triggers the alarm, wearable accessory, and sends a notification to the Web GUI

Software Components

- Database for employee contact information.

- The computer hosting the database changes depending on user configurations (Level of smoke to detect, Web GUI themes, alarm sounds).

- Database includes a table for user info (employee names, phone numbers, emergency numbers) and another table for sensor data (temperature data, smoke detection).

- Periodic timing loop is given by the smoke sensor monitoring the environment for smoke. The smoker sensor will log this data and send it to the web GUI every minute.

- The data the smoke detector logs will be displayed on the user website in the form of varying graphs and widgets.

- Smart watch will have a buzzer, Raspberry Pi with the alarm will have notification software. Raspberry Pi will send notifications to the Smart watch and to the web GUI.

- Bidirectional GUI included (user can reset the smoke detector from the web GUI, and changes made to the physical detector will show in Web GUI settings)

5 Additional Hardware Required

Developing FANS requires additional hardware that is described in the following tables. Table 3 lists hardware that is already provided by Carleton University, while Table 4 shows any additional components that will need to be purchased by the team.

Component	Amount	Digi-Key Number
Wires	50-100 pieces	N/A
SenseHat Module	4	N/A
Breadboard	4	N/A

Table 3: Additional hardware provided by Carleton University.

Component	Amount	Digi-Key Number
AudioHat Module	1	N/A
Smoke Sensor	1	333102
16x4 LCD Display	2	EA DOGS164W-A
Raspberry Pi Zero W	1	SC0065

Table 4: Additional hardware to be bought by the team.

References

[1] P. Matoušek, *Thick smoke on black background*. [Online]. Available: `https://www.freeimages.com/photo/thick-smoke-on-black-background-1633270` (visited on 02/11/2024).

[2] "Smoke alarms: A sound you can live with." (Sep. 29, 2020), [Online]. Available: `https://www.getprepared.gc.ca/cnt/rsrcs/sfttps/tp201011-en.aspx` (visited on 02/11/2024).

[3] (Oct. 7, 2022), [Online]. Available: `https://www150.statcan.gc.ca/n1/pub/11-627-m/11-627-m2022035-eng.htm` (visited on 02/12/2024).

[4] A. Erickson. (Sep. 15, 2023), [Online]. Available: `https://www.digitize-inc.com/blog/fire-alarm-whats-new-2023.php` (visited on 02/12/2024).

[5] (), [Online]. Available: `https://support.firstalert.com/s/article/Smoke-Alarms-while-Cooking` (visited on 02/12/2024).

[6] T. Preston-Werner. "Semantic versioning 2.0.0." (), [Online]. Available: `https://semver.org/` (visited on 02/12/2024).

[7] A. Oppermann. "What is the v-model in software development." (Apr. 6, 2023), [Online]. Available: `https://builtin.com/software-engineering-perspectives/v-model` (visited on 02/12/2024).

[8] "Getting started with the sense hat." (), [Online]. Available: `https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat` (visited on 02/12/2024).

[9] J. de Langen. "Sending emails with python." (), [Online]. Available: `https://realpython.com/python-send-email/` (visited on 02/12/2024).

A Something

This is a placeholder appendix.