



Bachelor's Thesis

Development of a Confined Spaces Dispatch Software for a Canadian Fire Brigade

by André Kuhlmann
2022-07-19

Faculty of
Electrical-Engineering and Information Technology

Supervisor: Prof. Dr. rer. nat. Wolfgang Lux
Advisor: Frederik Feichtmeier B.Eng.

Hochschule Düsseldorf
University of Applied Sciences

HSD

Fachbereich Elektro- und Informationstechnik
Faculty of Electrical Engineering and Information Technology



Abstract

This work focuses on designing an application ecosystem for a Canadian fire department. As it is part of the critical infrastructure, safety and code quality is of main concern. The thesis will go over the complete development life cycle, from planning to the actual implementation of a Computer-Aided Dispatch software for confined spaces.

It is looked into the Canadian legislation to derive application flows and data structure from it. At the end the reader will have a thorough understanding of confined spaces and how to come up with a working application infrastructure.

Index Terms

Software Development - NG911 - Confined Spaces

Contents

1	Introduction	1
1.1	Overview	1
1.2	Structure	1
1.3	Problem	1
2	Fundamentals	2
2.1	Computer-Aided Dispatch (CAD)	2
2.2	Campbell River Fire Department	3
2.3	Confined Spaces	5
2.3.1	Legislation	7
2.3.2	Regional Legislations	9
2.3.3	Standards	9
2.3.4	Hazard Diamond	10
2.3.5	Statistics	11
2.3.6	Training	11
2.3.7	Rescue	12
3	Capabilities	13
3.1	Terminal	15
3.2	Trigger	16
3.3	Dispatcher	17
3.4	Worker	17
4	Technologies	18
4.1	Electron	18
4.2	VueJS	19
4.3	Firebase	20
4.4	Version Control	21
4.5	Automated Testing	22
4.6	CI/CD	22
5	System Design	23
5.1	Application Flows	23
5.1.1	Register a new Contractor	23
5.1.2	Adding a Confined Space	24
5.1.3	Updating a Confined Space	24
5.1.4	Schedule Confined Space	24
5.1.5	Entering a Confined Space	25
5.1.6	Declaring an Emergency	26
5.2	Data Model	27
5.2.1	Contact	29
5.2.2	Users	30
5.2.3	Contractors	30
5.2.4	Confined Spaces	31
5.2.5	Operations	32
5.2.6	Capture Point	34
5.3	Infrastructure	35
5.4	Reliability	35

6	Implementation	36
6.1	Project Management	36
6.2	Interface Design	36
6.3	Software	37
6.3.1	electron-vite-fusion	37
6.3.2	cs-trigger and cs-terminal	38
6.3.3	cs-firebase-manager	39
6.3.4	cs-models	41
6.3.5	cs-dispatcher	41
6.3.6	cs-worker	49
7	Conclusion	50
8	Note of Thanks	51
	References	52

1 Introduction

1.1 Overview

In conjunction with the fire department of Campbell River I set out to create a software that would help them to remember and schedule operations in so-called “confined spaces”. Confined spaces being areas which are not designed for permanent human occupancy and therefore might pose a risk to human safety. In case of an emergency the staff is obligated to help with the rescue of any person inside that confined space.

1.2 Structure

The thesis is divided into several parts. Each of which builds up on the previous one. After giving an overview on what needs to be achieved and which features need to be implemented, I will go into more detail on how the whole system architecture has come to be.

1.3 Problem

As the old software never fully supported the capture of confined space operations a replacement is needed. By planning and developing a multitude of different applications the whole process from organization to rescue will be streamlined and improved. As the software is vital to perform, reliability and security will be of major concern.

2 Fundamentals

2.1 Computer-Aided Dispatch (CAD)

Computer-Aided Dispatch (CAD), not to be misunderstood with Computer-Aided Design¹ is a special category of software that's helping to dispatch emergency services such as police and fire stations [2]. The software is used to provide a common flow for handling incoming emergencies – to ensure a high agility and quick response to incoming emergency calls [3], [4].

More modern implementations of such a software are oftentimes referred to as **Next-Generation 911** (NG9-1-1). 911 being the common emergency number in North-America [5]. Whereas traditional dispatchment software was focused on communication via “static land-line connections” [6] NG9-1-1 focuses on providing advanced functionalities such as:

- processing of send video, audio and text messages [7]
- the ability to receive data from different “transmitting devices such as wearable medical devices, car computers and building alarms” [7]
- intercommunication between different departments and emergency services [7]

These features allow for better emergency service through new ways of communication and enable faster response times to incoming accidents.

What most of these software products have in common is a list of current emergencies and a map view that displays the location of any emergency units and the current incidents.

The fire department of Campbell River is currently using a CAD software by a company called Centralsquare (See Figure 1). [8]. Even though newer versions with more functionalities of that product do exist an older version is still used as the fire department is not directly in control of what software is allowed to be used on their official IT infrastructure.

¹**Computer-Aided Design** misleadingly also referred to as CAD-Software is a software that's main purpose lays in the creation of digital construction plans and three-dimensional (3d) models. It is typically used in mechanical engineering, civil engineering and electrical engineering. [1]

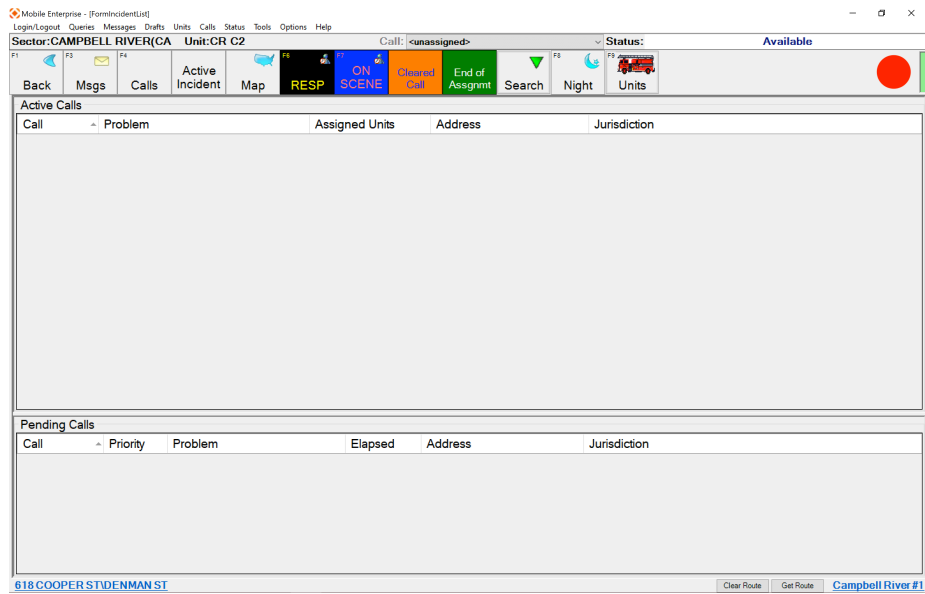


Figure 1: Centralsquare CAD - Source: K. Bellefleur [9]

2.2 Campbell River Fire Department

The city of Campbell River is located in the western part of Canada. It is the second-largest city on Vancouver Island. With a total population of more than 37,800 permanent residents [10]. The city is reachable by car, ferry, plane and seaplane and is about a 30-minute flight away from the metropolis of Vancouver.

The Fire Department is covering the wider area of Campbell River. The main fire station is located near the city center and houses a wide range of different fire trucks and rescue vehicles. [12]

A second, smaller fire station is located to the south of Campbell River and houses a second set of fire fighting vehicles [12].

Roughly 60% of all incoming emergencies happen to be medical calls, where the fire brigade is assisting with medical treatment of a patient. Approximately 10% of all responses are fire related and have to be extinguished. Besides these operations public assists, car accidents and false alarms are quite common. [14]



Figure 2: Map of Canada - Source: Mapbox [11], Author



Figure 3: 2015 Rosenbauer Commander 4000 - Source: S. MacKichan [13]

2.3 Confined Spaces

As defined by the Canadian Department of Justice, inside the **Occupational Health and Safety Regulations** (COHSR) a **Confined Space** is met by the following criteria [15] (11.01):

A space that...

- a. is enclosed or partially enclosed;
- b. is not designed or intended for continuous human occupancy; and
- c. has a limited or restricted means of entry or exit or an internal configuration that could complicate provision of first aid, evacuation, rescue or other emergency response.

A confined space can be classified as a **Hazardous Confined Space** when human health or life might be at danger due to one of the following conditions [15] (11.01):

- a. its design, construction, location or atmosphere;
- b. the materials or substances in it; or
- c. any other conditions relating to it.

A confined space is considered **Hot Work** if any source of fire or ignition is present. [15] (11.01)

Figure 4 shows a visual representation of the different kinds of confined spaces classifications.

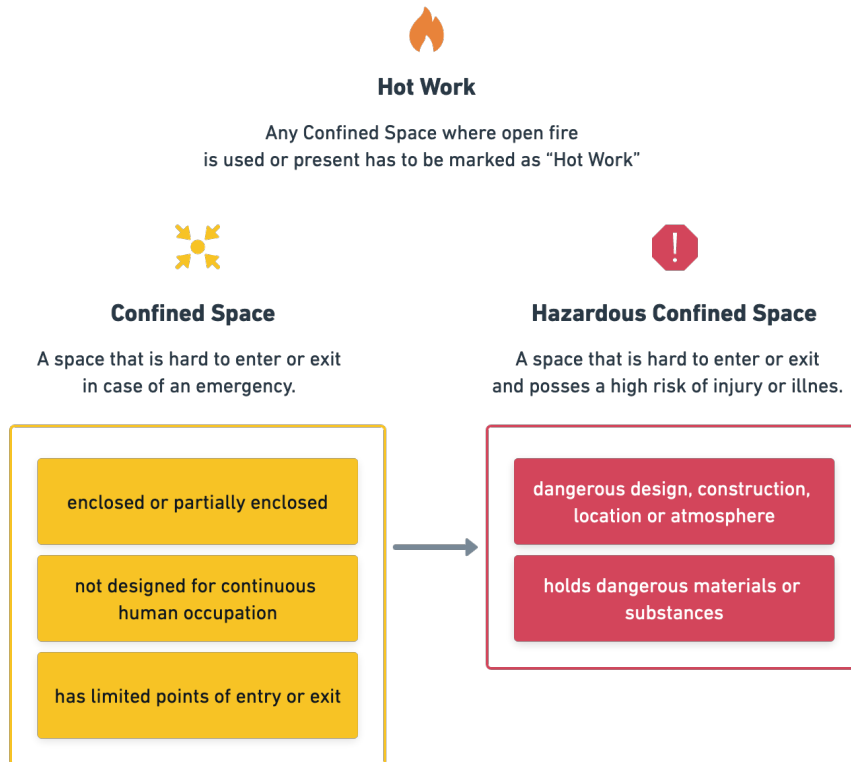


Figure 4: Confined Spaces Classification - Source: Canadian Department of Justice [15] (11.01), Author

Inside a confined space one might encounter one or more of the following hazards [16], [17]:

- **Physical Hazard** like *noise, engulfment, flooding, moving machinery, extreme temperatures, radiation, fire, explosions, ...*
- **Biological Hazard** like *mould, bacteria, viruses, blood-borne pathogens, biological toxins, ...*
- **Chemical Hazard** like *gases, vapors, fumes, chemical asphyxiant, lack of oxygen, asbestos, lead, ...*
- **Electrical Hazard** like *exposed wiring, electrical panels, ...*
- **Ergonomic Hazard** like *maintaining a difficult posture, repetitive strain, lifting, awkward reaching, use of hand tools, ...*
- **Psychological Hazard** like *working alone, claustrophobia, fatigue, ...*

Confined spaces are commonly part of a fabrication sites or public infrastructure and can be anything from a chemical tank to a sewage pipe. In order to protect workers entering these spaces, strict guidelines are put into place [14], [16], [18].



Figure 5: Hazardous Confined Space Entry - Source: Drägerwerk AG & Co. KGaA [19]

2.3.1 Legislation

Before a person is allowed to enter a confined space the employer first needs to perform, what is called a confined spaces identification and a hazard assessment, as described in subsection 11.02 to 11.03 of the COHSR [15]. From there on up a procedure for safe entry and exit can be established. Besides the COHSR each state builds up on these fundamental regulations and adds their own definition upon these.

Identification The identification process is a way of determining if there are any confined spaces on sight and whether they need to be classified as hazardous. Together with a qualified person, which is either someone from the work place committee or a health and safety representative the following actions are mandatory to be carried out:

1. The qualified person has to identify all confined and hazardous confined spaces.
2. The qualified person has to provide a catalog of all the confined and hazardous confined spaces that have been located in (1).
3. It is the employer's duty to properly sign these areas to prevent anyone from entering them.
4. The employer has to keep the record readily available. This document has to be updated / reviewed either every 3 years [15] (11.03) or whenever the conditions of that particular confined space change.
5. Before someone is granted access to a **none**-hazardous confined space. The employer needs to ensure that safety measures such as "person-check and emergency response systems" [15] (11.02(5)) are put into place.

The previously mentioned steps have been broken down to their core concepts. The full legislative definitions are part of subsection 11.02 of the COHSR [15].

Hazard Assessment For every confined space the assigned qualified person has to produce a detailed hazard report. The report has to include the following information [15] (11.03), [16]:

- The date at which the hazard assessment was carried out.
- An extensive list of all the hazards a person entering that space may encounter. Each hazard has to have a resolution on how a certain threat can be prevented, controlled or extinguished.
- It needs to mention which equipment is required on site by the worker entering the space and what equipment might be required by the first aid responders.
- What checks and tasks have to be performed before someone is allowed access to the confined space.
- It needs to clarify the way of getting approval to enter that space.
- Any additional information for the worker or the emergency responders has to be stated.

Only persons that have certain training and are qualified to perform a hazard assessment are allowed to fill out such a report [15] (11.03), [16]. In case of any accidents that have to do with a lack of information on that report the person whose job it was to create the report might be held responsible and might face legal consequences.

Just like the identification of the confined spaces has to be reviewed every 3 years and updated as soon as any changes happen to that confined space the hazards have to be reassessed as well. An exception being when the confined space was not entered in the previous 3 years and is not planned to be entered [16].

Establishing Procedures From the previous two reports a procedure for a safe entry and exit of a confined space can be derived. The procedures will set up a roadmap for all the actors involved. Every confined space needs to have “safe entry and exit procedures”, “two-way communication and person check systems” and “emergency response measures” [15] (11.04(2)(a)). In case of a hazardous confined space the procedure plan has to state which safety tests need to be performed ahead of time and what equipment has to be prepared for both the workers and the emergency response personal. [15] (11.04).

In case an entry permit system² is in place the qualified person has to fill out a permit handed out by the employer. A permit holds additional information about the hazardous confined space. [15] (11.04(3))

Depending on the type of hazard found in a confined space special rules and thresholds may apply. See subsection 11.05(1) of the COHSR.

²An **entry permit system** is a further safety system that includes a governing instance controlled by the employer. Its task is to schedule operations and hand out permits to the workers entering a hazardous confined space [3].

2.3.2 Regional Legislations

British Columbia (BC) is the 3rd largest state and also the most western state of Canada [20]. Additional to the laws and regulations published by the Canadian Department of Justice, through the COHSR the British Columbian statutory authority³ WorkSafeBC is responsible for concretizing these governmental laws.

Under subsection 9.37 ‘Provision of rescue services’ of the WorkSafeBC extended COHSR it is required, that any company has to supply a qualified rescue person on their own [22] (9.37(1)) or has to contract a 3rd party rescue service to perform a rescue if necessary [22] (9.37(2)).

The Fire Department of Campbell River provides this kind of service to local companies in their vicinity. Companies that claim their services have to sign a contract that enforces them to provide detailed information on their confined spaces described in *2.2.1 Legislation*. Before workers are granted access to perform work inside a confined space they have to provide a form to notify the fire department of their intended start and end time. In addition to that the form has to state possible hazards, the lockout board location⁴, the workers involved, information on entry and exit and the equipment required. Only after handing in that form and receiving the “get-go” from the fire departments operators – they are allowed to begin their work. [14]

2.3.3 Standards

A standard is a guideline established by a group of expert in a certain topic. Whilst regulations and governmental laws build the basis, standards ensures that certain tasks are performed with a high safety and a repeatable efficiency [24].

Common standards for operating a confined space in Canada are the CSA Z1006 and the NFPA 350 [17].

³A **statutory body** or **statutory authority** is a body set up by law (statute) that is authorized to implement certain legislation on behalf of the relevant country or state” [21]

⁴A **lockout board** or **lockout station** is a safety system that should prevent any hazards from happening by disabling and physically locking their functionality or control interface. When everything has been locked the key gets stored in a final container. Only the worker who is entering the confined space can remove all applied locks. That way it is made sure that no one can accidentally re-enable some functionality and cause an accident. [14], [23]

2.3.4 Hazard Diamond

In the NFPA 704 “Standard System for the Identification of the Hazards of Materials for Emergency Response” the association has constituted a sign for quickly identifying and assessing the potential risks from any substances or materials [25]. The sign is in the shape of a diamond and therefor often referred to as a “hazard diamond”.



Figure 6: Hazard Diamond - Source: NFPA - National Fire Protection Association [25], Author

The diamond is divided up into four different colored sections (see Figure 6), three of them showing a number with the associated risk level. Numbers range from 0 (no-risk) to 4 (highest-risk). Each color represents a different hazard:

- Blue: Health Hazard
 - 0: Normal Material
 - 1: Slightly Hazardous
 - 2: Hazardous
 - 3: Extreme Danger
 - 4: Deadly
- Red: Fire Hazard
 - 0: Will Not Burn
 - 1: Above 200°F ($\approx 93^{\circ}\text{C}$)
 - 2: Below 200°F ($\approx 93^{\circ}\text{C}$)
 - 3: Below 100°F ($\approx 37^{\circ}\text{C}$)
 - 4: Below 73°F ($\approx 22^{\circ}\text{C}$)
- Yellow: Instability Hazard
 - 0: Stable
 - 1: Unstable if Heated
 - 2: Violent Chemical Change
 - 3: Shock and Heat may Detonate
 - 4: May Detonate

Conversions from °F to °C were rounded down and have been taken from WolframAlpha [26].

The white section remarks a special area where hazards like

- ACID: Acid
- ALK: Alkali
- COR: Corrosive
- OX: Oxidizer
- W: Use no Water

can be denoted.

All previous hazard descriptions have been taken from Michael Mulders' blog posting [27].

2.3.5 Statistics

In Canada alone, the Canadian rescue training provider Roco Rescue recorded an average of 128 fatalities per year, in the timespan from 2011 to 2018 [28]. With 61% most casualties happen due to physical injuries such as falling objects and engulfment. 34% account to atmospheric hazards like “toxic chemicals, oxygen deficiency or combustible dust” [17].

As there are not a lot of statistics concerning injuries or fatalities in confined space, numbers may depend on the source of information.

In presentations by two engineers of 3M, they state that 60% of all casualties happen to would-be rescuers⁵ [17], [29]. These numbers go back to a study from 1986 and are rendered incorrect by an Australian study from 2019, which has taken multiple source from many developed countries to come up with their own incident estimations.

“In 1986, NIOSH issued an alert which concluded that ‘More than 60% of confined space fatalities occur among would-be rescuers’ (NIOSH, 1986). The sample size used to produce this alert was very small and had no stated selection criteria. [...] rescuer fatalities was much lower, and no more than 17% of all deaths” [30].

Because of the strict regulations put into place by the Canadian Government, deputy Chief K. Bellefleur can not remember a time when there has been any confined space accident in Campbell River in recent years. Most fatalities he says happen due to a lack of carefulness [14]. The problem with that being that training for this kind of rescue sometimes falls short as these events happen so infrequently if at all. As many of the abilities come down to well-trained risk assessment of their fire firefighters they will most likely be able to help a person in need, still there being a level of uncertainty when it comes to hazards that they not normally come across. [14]

2.3.6 Training

In order for someone to be allowed access to a confined space they need special certification from training providers. In courses workers and emergency responders will learn about entry and exit procedures, risk management, hazard assessment and hazard control. [31], [32]

About 90% of firefighters at the Campbell River Fire Department have undergone thorough training in Rope and Confined Space Rescue [14]. Because they are a comparably small fire department “everyone has to be good at everything” [14].

⁵**Would-be rescuer** Rescuer that died during the rescue operation

2.3.7 Rescue

In case of an emergency inside a confined space there are three types of rescue operations with increasing amount of complexity and risk [33]:

1. **Self rescue** Where entrants are able to evacuate the confined space on their own.
2. **Non-entry rescue** Entrants are unable to leave the confined space on their own but are able to be retrieved through an extraction system.
3. **Entry rescue** Entrants are unable to leave on their own and no extraction system is in place. A trained rescuer has to enter the confined space and recover the person inside.

Australian researchers Jason Selman, Jeffrey Spickett, Janis Jansz and Benjamin Mullis proposed a five-step rescue system they call **REALE** [33]. The system should provide a framework for emergency responders to ensure a well-organized sequence of events.

A citation of these steps from their publication ‘Confined space rescue: A proposed procedure to reduce the risks’ [33] states:

1. **Reconnaissance** of the rescue task
2. **Elimination** or reduction of hazards
3. **Accessing** the casualty, in which a minimal number of rescuers enter the confined space and make contact with the casualty.
4. Provision of **Life-saving** first aid to the casualty
5. **Extrication** of the casualty as required

3 Capabilities

Canadian authorities prohibited the management of confined spaces through the CAD software that is currently used at the Campbell River fire department. The planned software should fill in these gaps. As the original software was not designed to handle confined spaces, operators got creative and created pending emergency calls which should indicate that some workers would be in a confined space. [14]

The software should be rolled out in multiple release cycles with an increasing amount of functionality and complexity. As for the features required I have created a roadmap (Figure 7) that indicates the functionality implemented in each cycle.

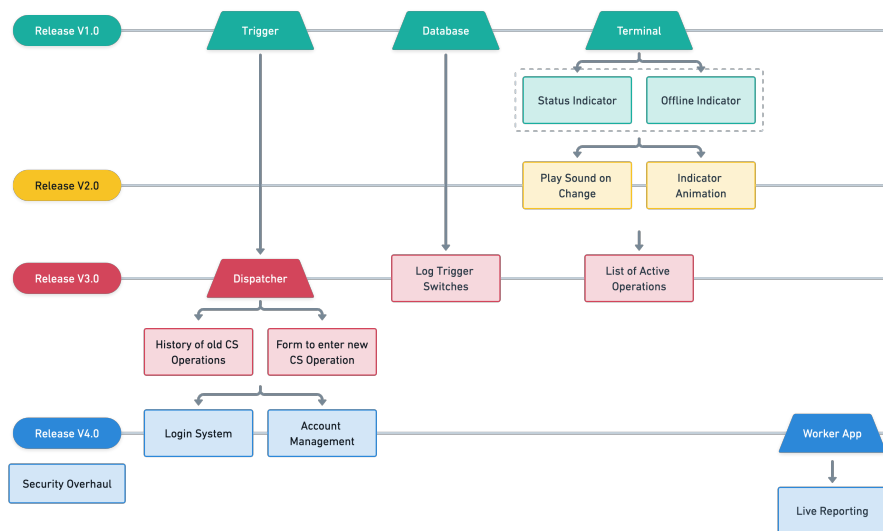


Figure 7: Software Release Roadmap

In order for the application to quickly be deployed into the field, the first two releases only consist of a very basic feature set to indicate whether there is an ongoing operation or not. With each release the application should evolve into a more complex and feature-rich application. Version 1 is the bare minimum of what is needed by the fire department. This would already drastically improve their safety concerning confined spaces, as they currently use sticky notes taped to the windshield of the vehicles to indicate that there is an ongoing confined space operation. [14]

The complete ecosystem should consist of multiple applications that should all handle a different task. In the following subchapters I will explain the purpose of each of the pieces of software and how they evolve throughout the different releases. I will list the requirements made by the fire department and add additional features and ideas that I came up with. See Figure 8.

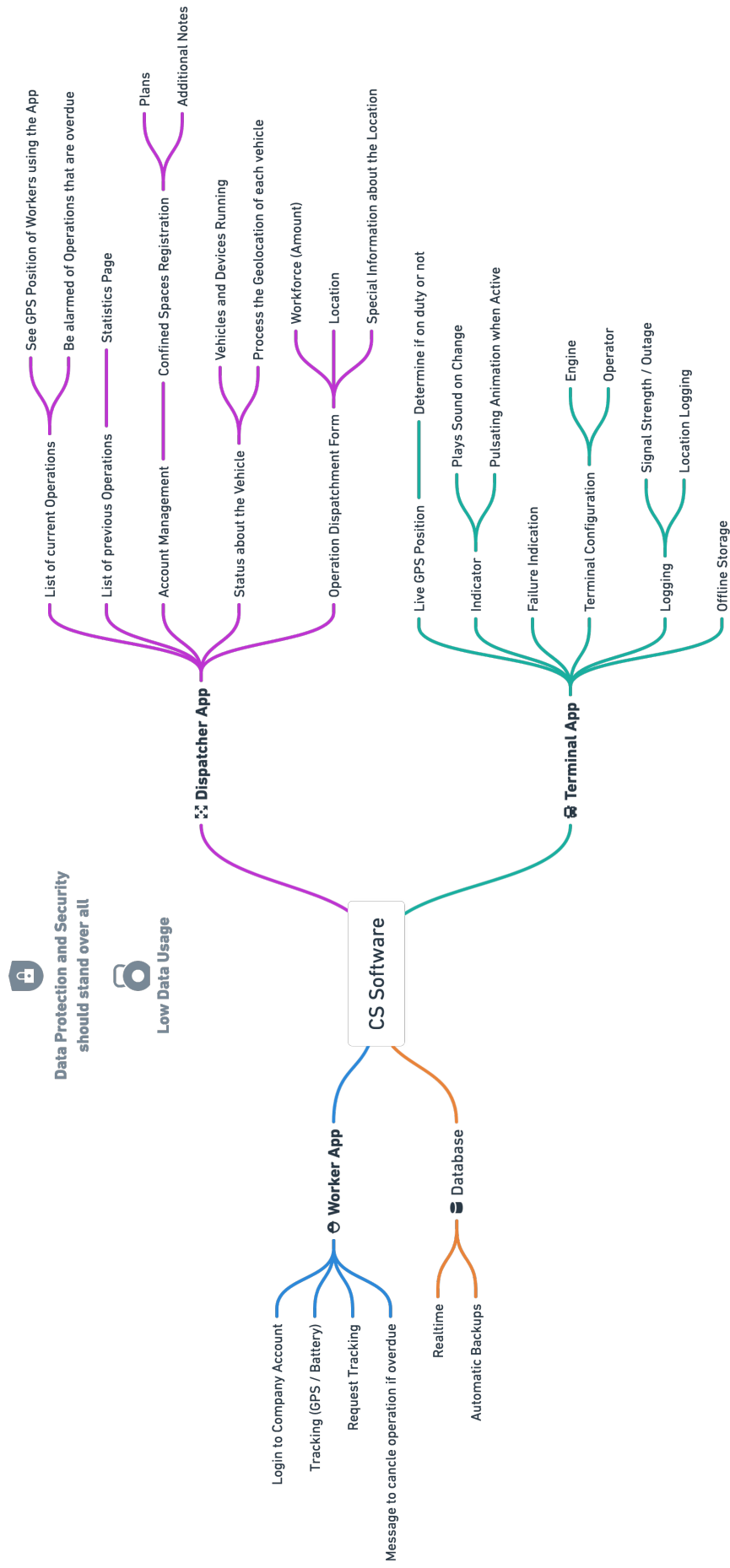


Figure 8: Feature Mind Map

3.1 Terminal

The Terminal application is deployed to the complete fleet of vehicles throughout the fire department. Each truck has one or more Windows based computers mounted to the dashboard or to the seats of the vehicle. See Figure 9.



Figure 9: Windows Tablet Onboard - Source: K. Bellefleur [9]

A small application, mimicking the functionality of a LED should indicate to everyone onboard that there is an active confined space operation. With that information the operators should be reminded that in case of any incoming emergencies they have to staff or call additional personal to standby for any incident happening inside the confined space. In case the incoming emergency requires the complete staff of the fire department the workers inside the confined space have to be notified and may need to terminate their operation.

Version 1 The first version of the application should have the very basic feature of displaying the on and off state with an additional offline indicator to give a warning whenever the data could not be retrieved from the backend.

Version 2 Building up on the first version. The terminal should now implement a sound whenever the state of the LED indicator changes and maybe play a little animation to subtly get the attention of whoever is looking at the monitor.

Version 3 As the application gets much bigger with the 3rd version of the ecosystem. More data will be provided by the operators. This will give the opportunity to display a list of confined spaces inside the vehicle.

Version 4 In case of an emergency the application should present an easy guideline on how to respond and visualize the key aspects of the confined space.

3.2 Trigger

This application runs on the operators desktop inside the dispatchment center. Whenever a worker calls in to enter a confined space and gets approval by the operators they are able to press a button that sends the status change to all running *Terminal* applications inside the vehicles.

As the software is not allowed on the main dispatchment devices this application will most likely run externally on a Microsoft Surface Pro tablet owned by the fire department. [14]



Figure 10: Dispatchment Center Campbell River - Source: NI911 [34]

The first version of both the *Terminal* and the *Trigger* build up the basis on what is required by the fire department.

Version 1 and 2 A simple button to toggle the confined space state on or off is more than enough. The state should be synchronized across all the applications.

Version 3 or higher Discontinued and replaced by the *Dispatcher* application.

3.3 Dispatcher

With the introduction of version 3 the *Trigger* will be replaced by a more feature rich application that should allow dispatchers to control operations with far more precision.

Version 3 The fire department could imagine the following features being implemented into the dispatchment application:

- Static input forms to input information about an operation, such as:
 - Company name
 - Site address
 - Site name
 - Confined space id
 - Hatchwatch⁶ name
 - Contact information
- A administration panel to see historic confined spaces operations

Version 4 This version should introduce some authentication process to protect the data from any unauthorized person to be accessed.

3.4 Worker

The *Worker* is a concept that I made an early prototype off. This application should introduce a new method for the worker to communicate with the fire department. I have imagined an app that can be installed on the workers phone. Inside the application – access to a confined space can be scheduled and requested from the fire departments operators without the need to phone in. When access is granted the phone will constantly send data back to the operators. In case the operation was not marked as completed by any of the workers and the predetermined exit time gets exceeded, operators will get informed over a possible issue and be able to check back on the workers well-being.

Possible data that can be transmitted to the operators might include:

- The GPS Location of the device
- Phone Acceleration Spikes
- Battery Level
- Connection Strength
- When connected to a smartwatch, send health stats such as the workers' heart rate

This would open up new ways of responding to emergencies and give the workers an extra level of security.

Although iOS usage on smartphones in Canada is quite high compared to Europe. The application should be developed for both major mobile operating systems iOS (Market Share Canada: 57.3% [35]) and Android (Market Share Canada: 42.5% [35]) to ensure maximum coverage.

⁶A **hatchwatch** is the person standing outside the confined space that is tasked with contacting the emergency responders in case of an incident.

4 Technologies

For the applications to work, a wide variety of technologies were used. This chapter should give a quick overview on each software and explain the core concepts needed to understand the further development of the applications.

A common application can be divided into two parts. The **frontend**, the applications that gets used by the end user and the **backend**, the applications that is used by the system itself. Most commonly the term frontend only refers to a website accessible through a browser, I will misuse this term for any native mobile application as both a native app and a website fulfill the same purpose – to provide an interface to the end user.

4.1 Electron

Electron is a development framework⁷ that enables developers to build cross platform⁸ desktop applications. Electron is based on Chromium and Node.js which both use the Google developed V8 browser engine⁹. Browser applications written in HTML, CSS and JavaScript can be run on any of the supported platforms. Different to a normal website the application does not need to be hosted on a server but instead gets downloaded and installed just like a normal executable would. Additionally, to using all the available APIs inside of Chromium, Electron comes with its own set of libraries for performing window manipulation or communicating with the host PC. [38]

The library can be installed via the Node Package Manager¹⁰ (npm).

```
npm install electron
```

Though Electron applications are quite large and resource extensive compared to natively compiled applications many big companies use it for their software. Microsoft Teams, Visual Studio Code, Discord, Slack, WhatsApp Desktop and many more all rely on Electron [39].

⁷A **framework** is a collection of libraries that abstracts some functionality and provides a starting point for all kinds of different application.

⁸**Cross platform** meaning that the same program can be run on different operating systems. [36], [37]

⁹A **browser engine** is used to render HTML documents, compile and run JavaScript code.

¹⁰A **Package Manager** is a piece of software that is normally provided with a programming language. It provides a way of installing libraries from a large repository of publicly shared software packages. Some examples of common package managers include: npm (Node.js), gem (Ruby), pip (Python), spm (Swift), pub (Dart), ...

4.2 VueJS

VueJS is a reactive frontend framework that is used to build complex web applications. It was developed and published by Evan You in February 2014 [40]. VueJS like other popular frontend frameworks such as React or Svelte is component based. That means that the final application is divided up into a multitude of different `.vue` component files. Each of these components holds its own state and logic [41], [42]. This enforces a reusable programming style and enables a faster design and development process.

```
<script setup>
const count = ref(0)
</script>

<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

The code snippet above shows a very simple VueJS component depicting a simple counter. The `count` variable is a reference to a number that is incremented each time the button is clicked. As soon as a change to the `count` variable is detected, the component will be re-rendered. This is what's called reactivity – where any input to the data automatically triggers the Browser to update. In vanilla JavaScript¹¹ the developer would need to handle the re-rendering of the component manually.

I choose VueJS as it has numerous developers who have created all sort of different libraries over the years. It is therefore one of the most popular frontend frameworks. Personally I like how VueJS functions as it is quite comfortable to read and write the components.

¹¹**Vanilla JavaScript** refers to JavaScript without the use of any Frameworks and simply relying on the built-in JavaScript features.

4.3 Firebase

Acquired by Google back in 2014 [43] Firebase is an App Development Platform [44] making it easy to build small and large scale applications. Their business case has grown over the years from a real-time database to an authentication provider, data storage provider, analytics and cloud computing service [44]. As of right now, their offering counts 18 products and services [43] to streamline the whole development cycle of a modern application. Official SDKs are available for all major platforms and programming languages.

Firestore Firebase offers two kinds of database services to their customers. Starting with the initial release of Firebase their Realtime Database Service was the default. With the introduction of Firestore in 2017 they made available a second solution which instead of using a JSON-Tree for storing data now uses a NoSQL database model. With the concepts of a document, collections and indexes this meant that queries were much faster and less cumbersome to implement. [45]

Authentication & Rules To restrict access to an application and to the database – Firebase Authentication provides an SDK for client-side authentication of a user. Multiple ways of authentication like e-mail and password authentication, magic-link authentication, 3rd party OAuth authentication and many more are supported. Through a custom domain-specific language [46] called “Common Expression Language” [47] each request will be checked against a pre-defined rule set [48]. “The rule set should follow the principle of least privilege” [49]. That means that any user by default should not have any privileges to read or write to the database. Access will then be given by whitelisting certain actions on certain documents.

In the example below any request made to the `/users/{uid}` document will get checked for the right permissions. Only users that are logged in (with a valid JWT token) will be able to **read** any document inside the `/users` collection. Through custom authentication claims a role based access control can be realized [49], [50]. In this example users belonging to the administrator role (role 0) are able to both **read** and **write** to any document inside the `/users` collection. Giving **write** access implies that the user is allowed to **create**, **update** and **delete** a certain document. Furthermore, users that are logged in are allowed to perform a **update**-request if the `/users` object matches the ID of the user itself `allow update: if matchesUID()`.

```

service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{uid} {
      allow read: if loggedIn()
      allow write: if isAdmin()
      allow update: if matchesUID()
    }

    // Checks if the user is logged in
    function loggedIn() {
      return request.auth != null
    }

    // Checks if the requested resource id matches the
    // uid of the caller
    function matchesUID() {
      return loggedIn() && request.auth.uid == resource.id
    }

    // Checks if the caller belongs to the admin group
    function isAdmin() {
      return request.auth.token.role == 0
    }
  }
}

```

Firestore Functions On many other cloud providers oftentimes referred to as lambda functions¹² Firebase offers a cloud service called Firestore Functions. By setting event triggers small operations can be executed without the need of a dedicated server infrastructure. A trigger might be an HTTP request to a predefined endpoint or a change inside the database. A full list of supported environments and languages can be found under: <https://cloud.google.com/functions/docs/concepts/execution-environment> [51].

4.4 Version Control

A version control system makes it easy to track changes made to a project. Over the years “Git has become the de facto standard for version control” [52]. In a 2021 survey by StackOverflow it ranked as the number one developer tool [53]. Git is a distributed version control system [54] and therefore allows for multiple users to work on the same project. Git is a free and open source software that was developed by the Linus Torvalds back in 2005 [55]. He originally created the project to better maintain the development of his ever-growing Linux operating system [55]. Git is a command line application and does not require a Graphical User Interface GUI. Every change gets tracked inside a special `.git` repository folder. Along with some metadata each change gets tracked and can be traversed to the original editor [52]. A Git repository can either be stored locally or on a

¹²In many programming languages so-called **lambda functions** refer to a concept of anonymous functions. In these languages functions can be assigned to a variable and be executed on demand.

remote server. You can set up your own repository server or use one of the many hosted services, like GitHub or GitLab.

GitHub calls itself “the largest and most advanced development platform in the world” [56]. The service comes with additional features to track issues, review and merge pull requests, and more.

4.5 Automated Testing

As the name “Automated Testing” implies automated testing scripts can be written to validate the operability and correct behavior of a software system. Instead of checking the functionality of a system manually on every change made to the codebase testing scripts will check defined conditions on their own. In practices like Continuous Integration (CI) and Continuous Delivery (CD) where code commits are pushed frequently to production automated testing ensures no bugs find its way into the deployment. [57]

In the most popular testing framework for JavaScript “Jest” a test might be written as such:

```
test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3)
})
```

Should someone change the behavior of the summation `sum` function the automated test will fail, and the developer will be notified.

4.6 CI/CD

In the philosophy of Continuous Integration (CI) and Continuous Delivery (CD) the development process is split into two phases:

- Continuous Integration: Is the process of verifying the quality of the code and checking the defined test cases.
- Continuous Delivery: The process of pushing the code to production.

Rather than using many feature branches inside a repository, which normally delays the deployment process due to many merge conflicts¹³, developers are encouraged to work as close to production as possible. Specially programmed pipelines will then handle the complete deployment process. Updates will happen at a more frequent rate hence boosting general productivity.

The process of developing these pipelines is referred to as Development Operations (DevOps). Through the use of CI/CD tools like Jenkins, Travis, CircleCI, GitLab CI or GitHub Actions the pipelines will be run automatically as soon as a change is pushed to the repository. GitHub Action uses a `.yaml` file to specify the operations to be executed. Inside this pipeline configuration file things like the branch or the operating system of the server executing the tasks is defined.

¹³A **merge conflict** occurs when two developers happen to work on the same lines of code. One of the developers then needs to manually evaluate the right logic and resolve the issue.

5 System Design

As the whole ecosystem is made up of many smaller software pieces discussed in the previous part *3 Capabilities* many design considerations had to be made. In this chapter I will walk over all the important concepts I have come up with and will go into more detail the more this chapter progresses. First I will lay out general concepts on how the application should function. Later I will get more specific and explain the individual software libraries and technologies used in order to accomplish the previously set goals.

5.1 Application Flows

For a good user experience the interactions with the software were designed before developing the applications. Each flow describes the way in which a fireman, an operator, a worker and an employer might work with the software to complete a certain task.

Version 1 and 2 are rather limited – their whole purpose is to simply signal and remind the fireman inside the fire trucks of any ongoing confined spaces operations. When approved by the operators they are able to press a button to notify all vehicles that there is a confined space operation running. As there are multiple operators inside the dispatchment center their triggering applications will also be informed about the status change.

Version 3 and above are way more complex. Following each functionality of the applications will be explained:

5.1.1 Register a new Contractor

Local companies are required by law to either have an emergency responder on their own or have a contract with one [14]. Each new contractor of the fire department will be able to get registered inside the system. When creating a new client inside the software a form should ask for relevant information about the work place. As there has been a huge gab in the detail of any contractors' documentation about their work sites [14] some information should be made mandatory to be provided, to ensure consistency across all confined spaces. The documentation of some ranged from complete folders of documents, hazards assessments to almost no information at all [14].

A copy of the contract and any other documents regarding the company should be uploaded to the system to remove the need for the operator to go through any folder and look for the documents. A physical copy should always be kept on hand in case of a system failure.

5.1.2 Adding a Confined Space

Before an operation can be scheduled the confined space needs to be registered. This process is done by the operator entering the information the contractor provided. The operator will be guided through this process to ensure all relevant information is captured.

- Name of the confined space
- Location of the entry point
- Lock-out board location
- Measurement taken before entry
- Hazards to be expected in the confined space
- Date of the hazard assessment
- Mitigation measures to be taken
- Equipment to be used
- Emergency equipment to be used
- Further documentation and files

After entering the confined space into the system a verification about the correctness of the information will be requested from the company through an automated E-Mail or SMS.

Should the date of the hazard assessment be older than the maximum 3 years – the operator is obliged to ask for an updated report. Shortly before the assessment is due the companies contact will receive an automated e-mail asking them for an updated report.

5.1.3 Updating a Confined Space

Besides adding a confined space the operator should be able to update the information about a confined space. This may be needed when some conditions inside the confined space changed. As with the creation of a confined space and the accompanied hazard report the contact person has to confirm the correctness.

5.1.4 Schedule Confined Space

The employer should be able to schedule a confined space operation. This should either be possible by calling the operator, handing in a filled out form or by using the *Worker* application.

Scheduling the operations requires some important information:

1. Which Confined Space is the operation going to take place?
2. When is the Operation going to take place?
3. Who to contact in case of an emergency?
4. Who is going to enter the confined space?
 - What is the maximum number of workers entering the confined space at any time?
5. What work is going to be performed?
6. Additional Information?
 - Hazards that could occur that are not covered by the confined space?
 - Rescue Equipment to the equipment that has been covered by the confined space?

Should the selected space's hazard report be older than the maximum age of three years a warning is displayed that the employer first needs to provide an updated hazard report, or he needs to reaffirm that the information is still up-to-date.

After filling out all the information a review page should give a summary of the data that has been entered.

Once the operation has been scheduled the emergency contact will receive an E-Mail or an SMS with the information provided in the form. By clicking on a link provided inside the E-Mail or the SMS the emergency contact can confirm the captured information is valid.

5.1.5 Entering a Confined Space

Any confined space operation has to be approved by the operators before someone is allowed entry. A request to enter a space can either be performed by calling in to one of the operators inside the fire department or by sending a request through the *Worker* application. Only if the confined space operation has been scheduled – an entry can be approved. This gives the fire department and also the workers the confidence that all precautions have been met.

Should the entry be requested through the *Worker* application, the operator will receive a push notification asking him to give approval. In the time the operator needs to respond the one requesting the entry will be shown a waiting screen warning him to wait for approval. In case the operator is not responding the worker will be told to call in to the operator.

Each operation can be in one of eight different states:

1. Scheduled
2. Requested
3. Canceled
4. Approved
5. Declined
6. In Progress
7. Emergency
8. Completed

Once a request to enter the confined space has been made to the operators the request will either be approved or declined. A decline of the request may be caused due to a lack of information about the operation or a lack of emergency responders inside the fire department.

Both the cancelation and the decline of an operation require additional information to why it was called off. Most predominant reason being a valuation of the risks being off limits. But might also be due to a rescheduling of the operation or a lack of workers.

Each step gets documented with the time and date of the event. This will provide essential knowledge to prove the correct execution of each of the steps.

Should the whole fire department be busy with a huge emergency requiring all staff the operators are able to decline all requests automatically and display a message to the *Worker* saying that operations are currently on hold.

5.1.6 Declaring an Emergency

Should there be an emergency inside a confined space all applications switch into an emergency mode. Fireman and dispatchers are shown clear instructions on how to react corresponding to the previously mentioned **REALE** [33] guidelines. Additionally, any registered hazards and their resolutions will be shown in a clear and understandable way.

An emergency can be initiated in a variety of ways:

- Someone makes a call to the fire department
- A person on sight uses the *Worker* application to trigger an emergency
- The *Worker* application has detected a sensor value exceeding a certain threshold
- The operation has not been marked as completed by one of the workers and the operation has exceeded the scheduled time limit

5.2 Data Model

For the whole application I have come up with a complex data model. I have tried to capture the most relevant information and created room for expansion where necessary. Inside the data model distinctions between the actual entity and nested type interfaces are being made. The nested objects provide further structure throughout the application, but unlike the entities these types are always stored inside an entity and therefore are not directly referenceable by any other object inside the database.

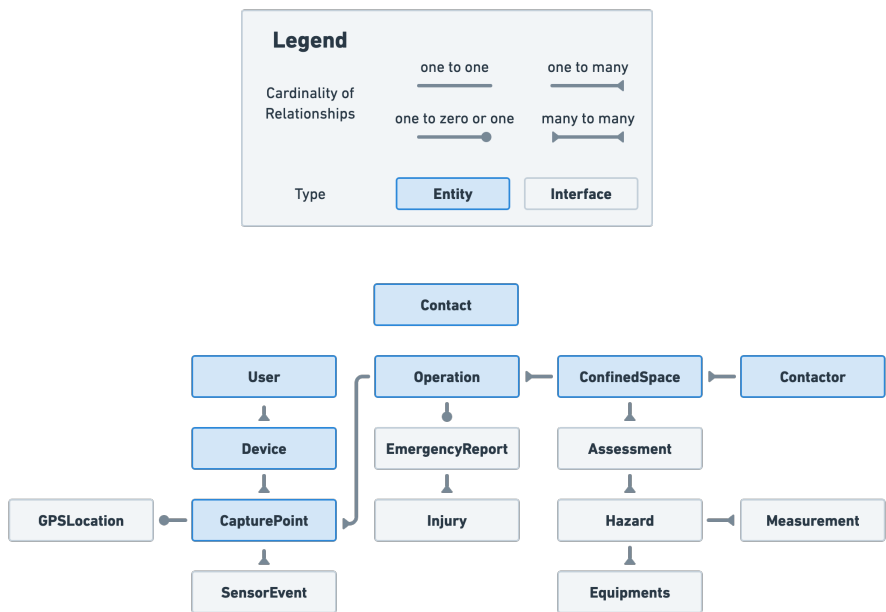


Figure 11: Condensed Entity Relationship Diagram

Each entity has a unique identifier (UUID)¹⁴ by which to reference the particular document. Adding to that – metadata information like a timestamp – when the entity was last changed `LastUpdated` and a reference to the user who performed that change `LastUpdatedByUser` are added to each entity. The names are purposely chosen to only reflect the latest change. Any manipulation to a document gets stored inside a subcollection called `History`.

To keep data usage at a minimum the `History` subcollection does not store complete snapshots of the object but only holds the changes that have been performed on the previous data snapshot. Besides the actual change made to the object each change-document will always include the previously mentioned `LastUpdated` and `LastUpdatedByUser` fields. The Internet Engineering Task

¹⁴A **Universally Unique IDentifier** or short **UUID** is generated from a set of five different algorithms which produce a 128 Bit long string that helps with unique labeling of data for better “sorting, ordering, and hashing of all sorts, storing in databases, simple allocation, and ease of programming in general”. [58]

Force (IETF)¹⁵ put out a Request For Comments (RFC)¹⁶ that describes how one might store changes made to a JSON object. The standard has been published in April 2013 under the name “JavaScript Object Notion (JSON) Patch” (RFC 9602) by Paul C. Bryan and Mark Nottingham. [63], [64]

It has been adapted by many existing libraries for all major programming languages [64]. It is therefore fairly straightforward to implement into the application.

As an example for the following object...

```
{
  "foo": "abc",
  "bar": "def"
}
```

... a change using the RFC 9602 standard might look like this:

```
[
  { "op": "replace", "path": "/bar", "value": "abc" },
  { "op": "remove", "path": "/foo" }
]
```

Each change inside the array describes one of 6 different case-sensitive operations that can be performed on the original data object. The operations `op` are:

- **add**: Add a new property to the object.
- **remove**: Remove a property from the object.
- **replace**: Replace the value of a property.
- **move**: Move a property to another location.
- **copy**: Copy a property to another location.
- **test**: Test that a property exists and has the expected value.

Only the **add**, **remove** and **replace** operations are of interest to this application.

With all of that information a history graph can be created and traversed to the user that created or modified the object at any point in time.

¹⁵The **IETF** or *Internet Engineering Task Force* is a standards body that focuses on developing and publishing standards for the open web. [59]

¹⁶**RFC** stands for *Request For Comments* and describes a standard published to the IETF. It is called *Request For Comments* as a standard is not directly recommended by the IETF and requests to be evaluated by anyone [60], [61]. It first has to go through a number of stages before it should be used in production by anyone. [62]

5.2.1 Contact

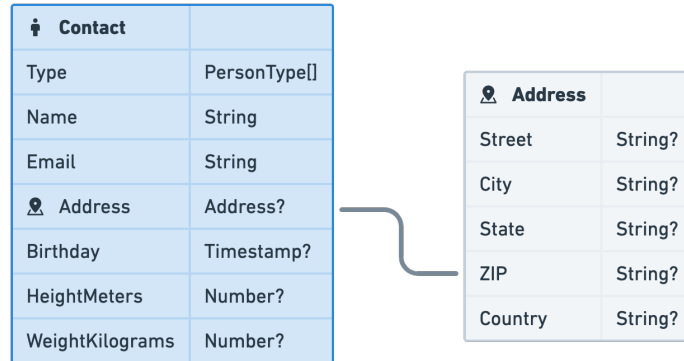


Figure 12: Contact Entity

A contact holds relevant information about an actor involved in the confined space process. The entity stores data like the persons name, address, phone number, email address and a selection of types to better identify the duty of the person. Each contact has a few mandatory fields and additional information like the persons body measurements which are optional¹⁷ – symbolized in Figure 12 by the ? question mark. These fields will most likely not be provided and are a demonstration of where additional fields might be implemented. Any frontend application accessing the data needs to be able to handle the absence of these values accordingly.

As the contact information plays an important role for the traceability of information the contact object is not allowed to be deleted. The contact entity in Figure 11 shows no connection to any other objects as the contact entity can be found all throughout the model structure.

¹⁷Present in many modern programming languages an **optional value** is one that can deliberately be omitted. The absence of a value is often denoted by a special **undefined**, **nil** or **null** value. A type annotation often uses a ? to indicate that the value is optional. [65], [66]

5.2.2 Users

 User	
 UID	UUID
Role	UserRole
Contact	Contact
Description	String?
LastLogin	Date?
Deleted	Date?

Figure 13: User Entity

Every person that wants to interact with the application has to be a registered user. Users have a role assigned to them. The `UserRole` field determines what operations the individual user is allowed to perform and which parts of the application he is allowed to access.

5.2.3 Contractors






 Contractor	
 ID	UUID
Name	String
 Spaces	Ref(ConfinedSpace)[]
Documents	Document[]
 Contacts	Ref(Contact[])
Description	String
 Workers	Ref(User)[]

Figure 14: Contractor Entity

Any institution that wants the fire department to be their emergency responder for any of their planned confined space entries has to sign a contract assuring the fire department that they will provide them with accurate information about their work place and the hazards that are to be expected inside their confined spaces.

The contractor entity holds general information, like the companies name and information about whom to contact for any administrative or technical issues related to the work place. Furthermore, the contractor entity holds references to their confined spaces and their workers' user accounts. More on that later.

Inside the `Documents` property the fire department is able to store any documents that are relevant to the work place, like the signed contract, maps, floor plans, etc. As for Firebase – any files that have been uploaded will get stored inside the Firebase Storage service. The property inside the database document will then hold a URL to access the particular file.

5.2.4 Confined Spaces

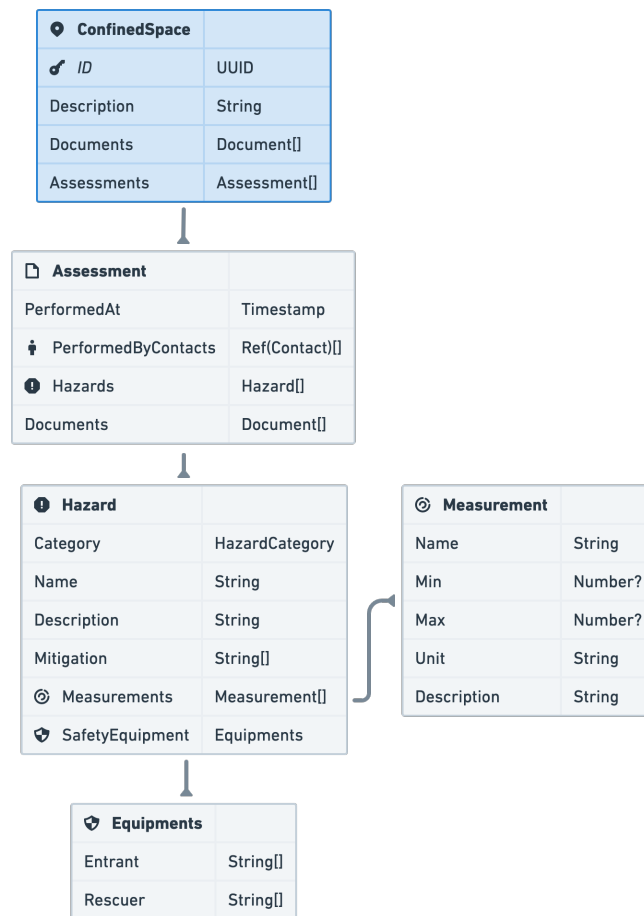


Figure 15: Confined Space Entity

A confined space builds a vital part of the application. Each confined space object is referenced by a single contractor. As mentioned in 2.3.1 (*Hazard Assessment*) the contractor has to conduct an assessment of the hazards that are to be expected inside the confined space.







Assessment For each confined space there are subsidiary assessment interfaces holding the information about who performed the assessment, a timestamp when the assessment took place, further documents and an array of hazards found inside that confined space. (See Figure 15)

Hazard In case of an emergency a hazard interface gives clear instructions on what to be expected when arriving at the scene. It states threshold values inside the `measurements` property that need to be checked before entry and potential mitigation strategies that have to be carried out. A possible hazard object might look as follows:

```
{
  "category": "physical",
  "name": "Extreme Temperatures",
  "description": "Inside the machinery compartment temperatures
    might exceed 50°C.",
  "mitigation": [
    "Turn off the machine.",
    "Open up emergency cooling vents."
  ],
  "measurements": [
    {
      "name": "Temperature",
      "min": null,
      "max": 65,
      "unit": "°C",
      "description": "The temperature inside the machinery
        compartment."
    }
  ],
  "safetyEquipment": {
    "entrant": [
      "Drinking Water to prevent hidration",
      "Thermostate with an overheating alert",
    ],
    "rescuer": [
      "Heat resistive suite"
    ]
  }
}
```

5.2.5 Operations

Each operation inside a confined space has to be announced by the contractor at least 24 hours prior to the start of the operation. Information inside the object includes the space where the operation is going to take place, the scheduled start and end-time of the operation, a description of the work that is going to be performed, the workers involved in the operation. The optional `ApprovedByUser` field gets filled by an operator as soon as he gives the get go for the operation. This field is only to be set 30 minutes before the scheduled start so that approval can only be gained judging the current workload of the fire department. For further documentation an optional `StartedAt` and `EndedAt` date can be used to track the actual start and end of the operation. Not mentioned rescue information can be specified in a string field called `RescueInformation`.

 Operation	
 <i>ID</i>	UUID
 Space	Ref(ConfinedSpace)
ScheduledAt	Timestamp
ApprovedAt	Timestamp?
ApprovedByUser	Ref(User)?
StartedAt	Timestamp?
EndedAt	Timestamp?
 Entrants	Ref(Contact)[]
 Attendees	Ref(Contact)[]
Description	String
LockOutBoardLocation	String?
RescueInformation	String?
 Emergency	EmergencyReport?
Documents	Document[]

 EmergencyReport	
 Rescuers	Ref(Contact)[]
 Injuries	Injury[]
Type	ResuceType?
Description	String?
Documents	Document[]
RecoveryStartedAt	Timestamp?
RecoveryEndedAt	Timestamp?


 Injury	
Description	String
Person	Contact
DueTo	Hazard[]

Figure 16: Operation Entity

Emergency Report In case of an emergency each operation can have its own emergency report. The object is optional and can be used to further document what happened during the operation and what lead to the emergency.

5.2.6 Capture Point

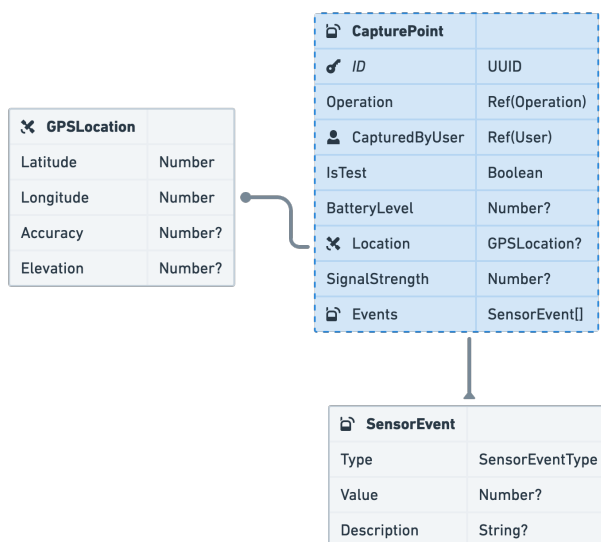


Figure 17: Capture Point Entity

This entity enables some of the smart features that should set this software apart from other software packages and enables the NG9-1-1 aspect of the application. Any entrant using the *Worker* application has the ability to transmit live data back to the fire department. In a predetermined interval the app will send status updates that include the users' location, the battery level and the signal strength. Should a device be offline the data will be stored on device with a signal strength of 0 and transmitted as soon as the device is back online again. Should the workers phone stay offline for a long time the operator will get notified that no device data is coming in.

Integrated into the capture point is an enumeration of different **SensorEvents** that enables the device to send special messages back to the *Dispatcher*. One of these special messages is the **SensorEvent.DeclareEmergency** which will immediately switch all applications into an emergency mode. For testing purposes a special testing flag **IsTest** can be set. This will prevent the applications from reacting to any of the sent data. For the initial release there are five **SensorEvents** implemented into the application:

- **DeclareEmergency:** An immediate emergency is triggered. Either the worker pressed the emergency button or the device considers there to be an emergency.
- **HighAcceleration:** The phone or the connected smartwatch has experienced a high acceleration spike which might be an indication for an accident.

- **HighHeartRate**: A connected smartwatch has detected a high heart rate.
- **LowHeartRate**: A connected smartwatch has detected a low heart rate of the worker.

In the future the application should be able to communicate with on-sight sensors to further monitor any dangers.

- **MeasurementExceeded**: A predefined threshold has been exceeded.

5.3 Infrastructure

There are two ways of how the whole application infrastructure can be set up. Both ways come with their advantages and disadvantages. One might choose a fully managed service like *Firebase* which has strong security guidelines against potential attackers though raises some data privacy concerns. The other way is to set up a dedicated server that can be customized to your needs, this may eliminate most of the privacy concerns at the cost of a far more complex setup and maintenance process.

To save time I have gone for the fully-managed solution as I am quite familiar with Firebase and have been using it across many of my projects. For transparency users would need to be informed that the application is hosted on Google's Servers. Firebase offers an extensive free tier that should fit the needs of this application just fine, but it has to be kept in mind that a fully-managed solution is most likely much more expensive at scale than a custom solution.

Purpose of many NG9-1-1 applications is to provide interoperability between different emergency services, as this is not the focus for the application that aspect has been discarded. In case multiple fire departments wanted to use the software a Terraform¹⁸ script could be created to quickly spin up new instances of the application.

5.4 Reliability

For an application to be reliable it has to be thoroughly tested. In the case that there is an error either in the communication infrastructure or a bug inside the software the application has to be able to recover from it or at least provide the user with a clear error message.

For all the applications I have used a service (Sentry) that automatically generates error reports and captures them inside a detailed list telling me what went wrong.

¹⁸**Terraform** is an *Infrastructure as Code* tool that allows for programmatically defining and setting up server infrastructure. [67]

6 Implementation

Besides conceptualizing the application, I have fully developed the first two versions of the applications. As time ran out at the end – the 3rd and 4th versions are still in development and need some refinement to be completed. In the following sections I have documented all software I have implemented so far.

6.1 Project Management

As time for this thesis is limited a strict project management strategy was needed. In a process called Kanban every task was broken down into smaller more manageable steps that provided a clear roadmap for each of the releases. In the end time was at a premium, so I decided to switch to a more agile development approach. Which meant that I had to implement both version 3 and 4 at the same time to get some presentable results.



Figure 18: Kanban Board

6.2 Interface Design

To speed up the development process most of the User Interface I have designed beforehand in a user interface design application called “Figma”. I have tried to come up with a unique and intuitive design that is easy to understand. Colors are kept at a minimum to not distract from what is most important to the

person using the application. Examples of the interface are shown in Figure 19 and Figure 20.

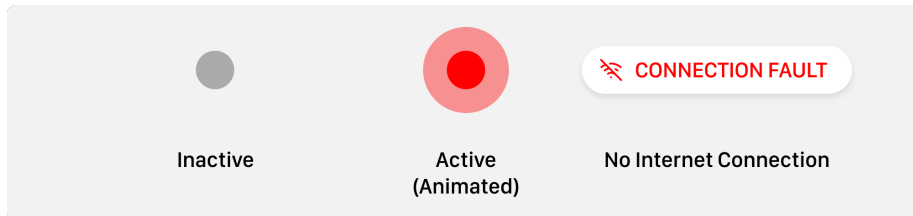


Figure 19: Figma Terminal Indicator States

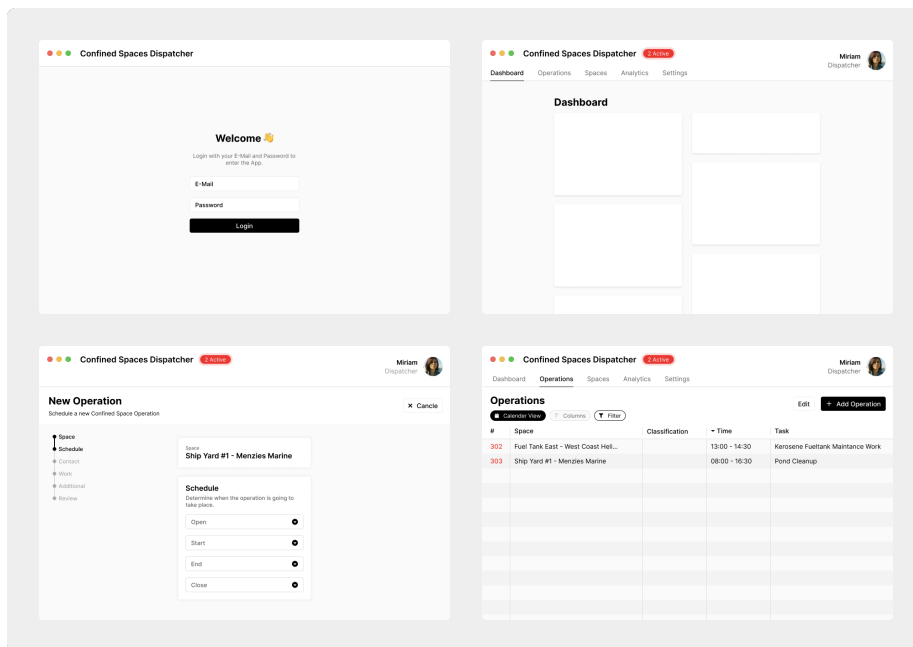


Figure 20: Figma Dispatcher Mockup

6.3 Software

Following each software implementation is explained in greater detail. The chapters are named like their repository counterparts.

6.3.1 electron-vite-fusion

As multiple applications relied up on Electron as their application core I first created a boilerplate to more quickly get started with writing the actual applications. The boilerplate which can be found under <https://github.com/KuhlTime/electron-vite-fusion> includes a build tool called “Vite” [68]. Depending on the environment the application is run in (development/production) the Electron app either servers content from a local development server or uses the



Figure 21: Electron Vite Fusion Logo

distribution build which includes the compiled VueJS application. The development server has hot module replacement enabled so that the application can be reloaded without a full reload of the actual webpage shown inside the Electron window. A package called “electron-builder” was used to automatically generate the binaries to be installed on the target platforms.

6.3.2 cs-trigger and cs-terminal

Based on the design I have created in Figma and the *electron-vite-boilerplate* the application is fully operable and can be used by the fire department. A continuous deployment action on GitHub is responsible for automatically deploying the application to the target platforms (macOS, Linux, Windows) and providing GitHub with the executables which are made available to download through the releases’ section of the repository.

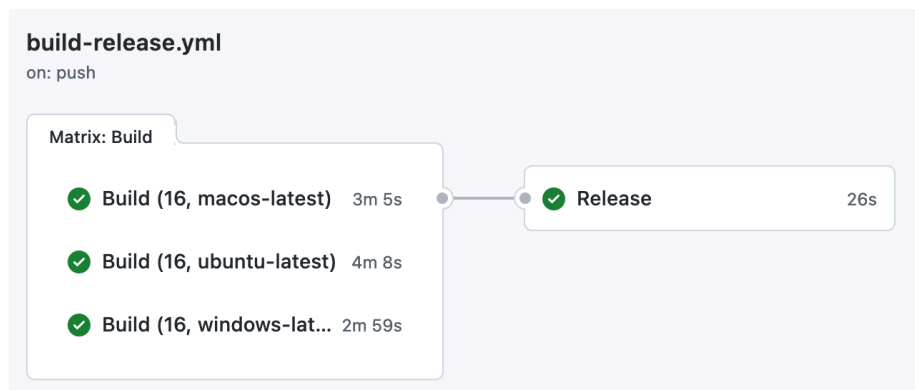


Figure 22: GitHub Actions Electron Deploy

As seen in Figure 22 the application has to be compiled for each target platform individually. For that GitHub Actions can define a matrix of options an action can be executed with.

```
jobs:
  build:
    strategy:
      matrix:
        os:
          - macos-latest
          - ubuntu-latest
          - windows-latest
    runs-on: ${{ matrix.os }}
    #...
```

The build job will therefor be executed on three different runners (servers).

6.3.3 cs-firebase-manager

This repository holds the security rules for the Firestore Database as well as the Firebase Functions that are to be executed. On each push to the repository two GitHub Action workflows are run to automatically deploy the code to Firebase.

Functions As of now there are two lambda functions enabled. The first will synchronize the settings of the Firestore users document with the user data in the Firebase Authentication service. The second lambda function can be triggered by an HTTP POST request made to the `/createNewUser` endpoint. The function checks that the user making the request belongs to the Administrator role. If so – both a Firebase Authentication User and a new Firestore document with the users unique identifier `uid` is created.

On creation a new user is assigned a randomized token generated using the `uuid` NPM package. The user will then be sent an email with a link to the registration page and the generated token as a query parameter.

https://crfd.ca/verify?email=andre@kuhlti.me&token=some_random_token

For the email that is sent out to a new user I am using a service called Postmark [69] which allows one to sent out 100 emails per month for free. Using email templates the emails can be pre-styled and filled with content by naming the variables in the template. A variable might be used like follows: `<p>{{{example_variable}}}</p>`.

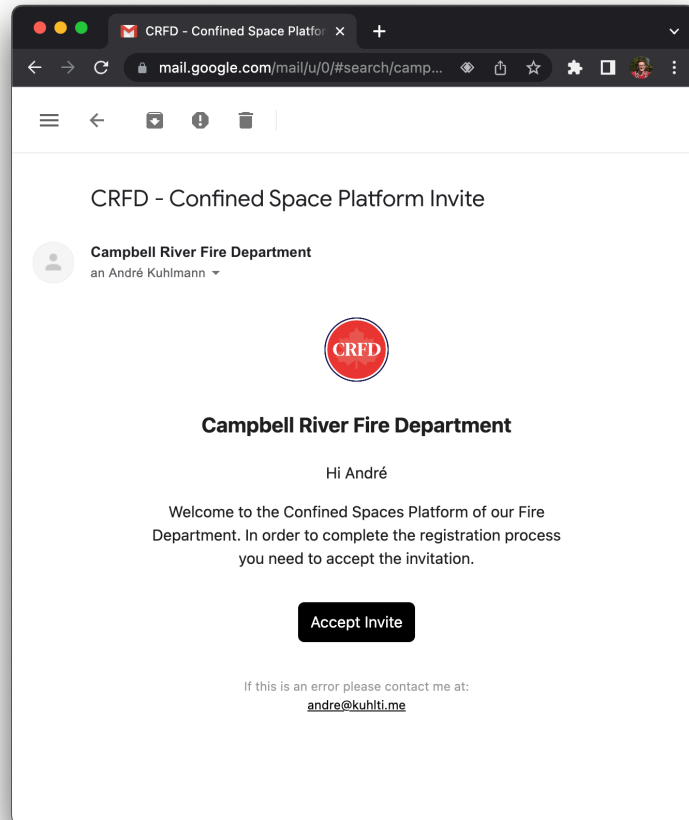


Figure 23: Registration E-Mail

Security Rules The security rules are checked against a large set of test cases that all need to be passed. These test cases all try to perform CRUD¹⁹ operations on different documents inside the Firestore database. The tests can either be run on device using the `npm run test` command or get executed automatically when the rules are pushed to the GitHub repository. In order for Firebase to test the rules an emulation service needs to be executed. This will spin up a fake Firebase emulator inside of which the rules can be tested without causing any harm to the real Firebase service. The action will be executed on a Linux machine and contains the following steps:

1. Clone the repository to the runner instance
2. Setup Node.JS
3. Install the NPM packages
4. Setup Java (needed for the emulator)
5. Install the Firebase Emulator Suite
6. Start the emulator
7. Perform the tests
8. If tests passed: Deploy the rules to Firebase

6.3.4 cs-models

All applications are written mostly in TypeScript or JavaScript. To ensure interoperability between the different applications I have created a NPM package that can be easily imported and used inside all applications. The package is automatically distributed to the official NPM registry and can be installed by running the following command:

```
npm install @crfd/cs-models
```

On every version tag assigned to a commit the package is then automatically bundled and uploaded by a GitHub Action workflow. The workflow installs all packages and runs the TypeScript compilation. The upload is then handed over to a third party action. In order to sign in to the correct NPM account an API token `NPM_TOKEN` has to be provided to the action. The token is stored inside the repository secrets.

```
- name: Publish to NPM
  uses: JS-DevTools/npm-publish@v1
  with:
    access: public
    token: ${{ secrets.NPM_TOKEN }}
```

6.3.5 cs-dispatcher

Instead of using an Electron window to display the dispatchment application. I have gone for a complete web solution. The application is hosted on a server that is accessible from the internet. Changes to the repository are automatically being uploaded to the server.

¹⁹**CRUD** stands for **Create**, **Read**, **Update** and **Delete** and are different kind of operations that can be performed on a database

Equally, to all the other UI applications I am using VueJS as the frontend framework. As this is the largest application of them all, multiple pages are of need. For this I used a VueJS standard library called “vue-router”. Instead of providing each endpoint with its own HTML page, every request made to the website is automatically forward to the index page of the website. This is called a single page application as there is only one page that is effectively being displayed. Every route change is captured by the vue-router and triggers an on-page component to be loaded inside the `<router-view />` tag. Figure 24 shows the difference between a “normal” website that does not use a single page for its content (left) and the vue-router (right).

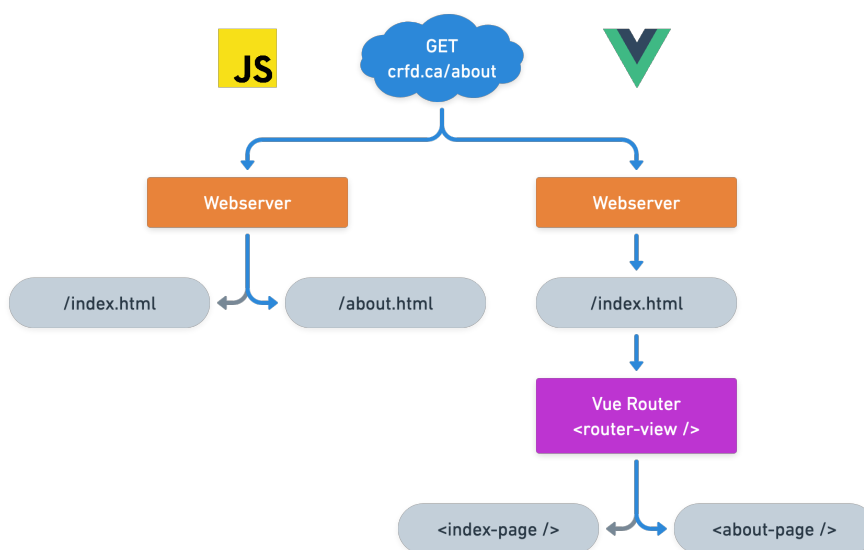


Figure 24: Vue Router

All routes are defined in a settings object and passed to a new instance of the router. For manipulating the UI or restricting access to certain pages each route has special metadata object to define additional properties. The code snippet below shows the definition of a single route.

```
{
  path: '/',
  name: 'home',
  component: () => import('@pages/Dashboard/Home.vue'),
  meta: {
    title: 'Home',
    requiresAuth: false,
    hidden: false,
    icon: HomeIcon,
    hideFromCommandPalette: false
  }
}
```

Before navigating to a new page the router checks if the route requires authentication. If so, the user is only granted access to the page if he has been successfully logged in, if he is not logged in he will automatically be redirected to the login `next({ name: 'login' })` page. If the user is logged in or the page he is trying to access does not require him to be logged in, the route is allowed to be accessed `next()`.

```
const authGuard = (  
  toRoute: RouteLocationNormalized,  
  fromRoute: RouteLocationNormalized,  
  next: NavigationGuardNext  
) => {  
  if (toRoute.meta.requiresAuth) {  
    isAuthenticated() ? next() : next({ name: 'login' })  
  } else {  
    next()  
  }  
}
```

Because the source code on a website is publicly accessible and can be manipulated the routers' authentication guard is able to be circumvented by a potential bad actor. The attacker is though not able to gain any data, because fetching data from the Firestore database would require him to be a valid user that checks out against the Firebase's security rules (JWT token is invalid).

All components used by the application have been custom designed inside of Figma and then realized with VueJS. Here are some of the components used inside the application:



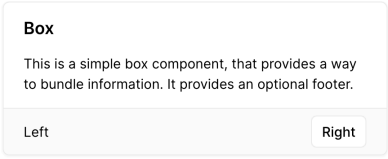



Component	Image
<p>Avatar: Displays the user's profile picture and name. The component takes the users <code>name</code>, <code>image</code>, <code>email</code> and the <code>avatar-size</code> as arguments. Should no <code>image</code> be set the <code>email</code> is checked for a Gravatar image. The <code>avatar-size</code> is used to define the size of the image. Depending on the size – on hover the full-name or the initials get shown.</p> <p><i>Left:</i> No Image Found <i>Middle:</i> Gravatar <i>Right:</i> Custom Image (Hover)</p>	<p>Normal Avatars:</p>  <p>XL Avatars:</p> 
<p>Box: A Box component is used to provide a common interface throughout the application. Each Box has a footer to display any state information on the left and buttons on the right. State information may be some validation error or a success message when the boxes content was executed without errors.</p>	
<p>Button: Buttons inside the applications come in 6 different flavors. Critical operations are denoted with a bright red color and all uppercase text. Optionally an icon can be set that is displayed in front of the buttons text.</p> <p><i>Left:</i> Button with Icon <i>Middle:</i> Button without Icon <i>Right:</i> Button without Icon (Disabled)</p>	<p>Primary:</p>  <p>Secondary:</p>  <p>Critical:</p> 

Table 1: *Dispatcher* Component Library

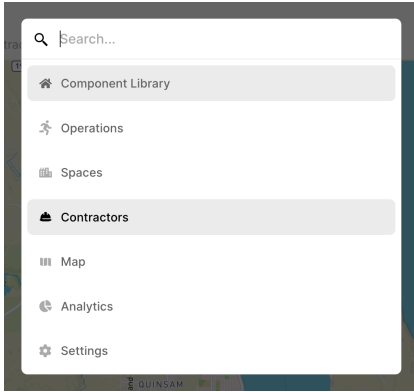
Command Palette: The Command Palette is accessible throughout the whole application either by pressing the search button in the navigation bar or by pressing the hotkey **CMD+K** or **STRG+K** and provides an easy way to jump between pages or execute common actions. Each command has its own **Command** typescript model. Inside the model exists a **name**, an **icon** and a **action** property. The latter is holding a lambda function which gets executed when the user selects the particular command. Extending on the **Command** model are different command categories like **RouteCommand** that takes a vue-router route as a variable. Objects like **Spaces** **Contractors** **Operations** stored inside the Database can be searched for as well.

When the command palette opens up the search field is automatically selected. When entering a search string the list of available results will get filtered.

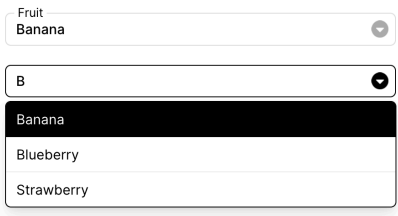
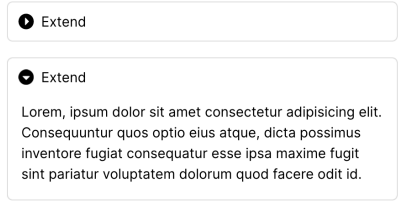
Detail: Custom styled detailed box. Using a VueJS `<slot></slot>` element the content inside the detail can be customized.

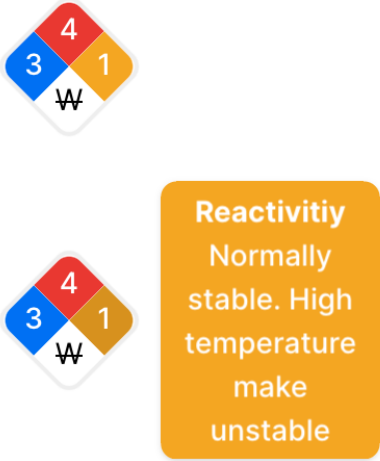
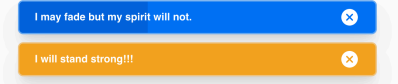
Select: The selection element is a rather complex component that allows a user to select from a list of provided options. He can either scroll through the list of possible options or he can enter a value to filter the shown list. To make the experience more pleasant the element has some saddle animations.

Command Palette:



Navigation Bar:



<p>Hazard Diamond: To more easily identify the hazards present inside a confined space a special hazard diamond (NFPA 704) component can be used. On hover the user can see a quick description of the present hazard and the respective risk level displayed in one of the four quadrants.</p>	
<p>Message: There are two types of messages that can be displayed. One that is auto-dismissed when the message is just used to inform the user over some none critical process, or it can be permanent to only be dismissable by the user clicking the dismiss button. All messages are overlaid over the applications content in a special <code><message-center /></code> component.</p>	
<p>Path: To display progress a multitude of different steps to better guide the user.</p>	<ul style="list-style-type: none"> ● General ● Hazard Assessment ● Contact ● Documents

To view the application and its components yourself visit dispatcher.kuhlti.me.
Following are some screenshots from within the application:

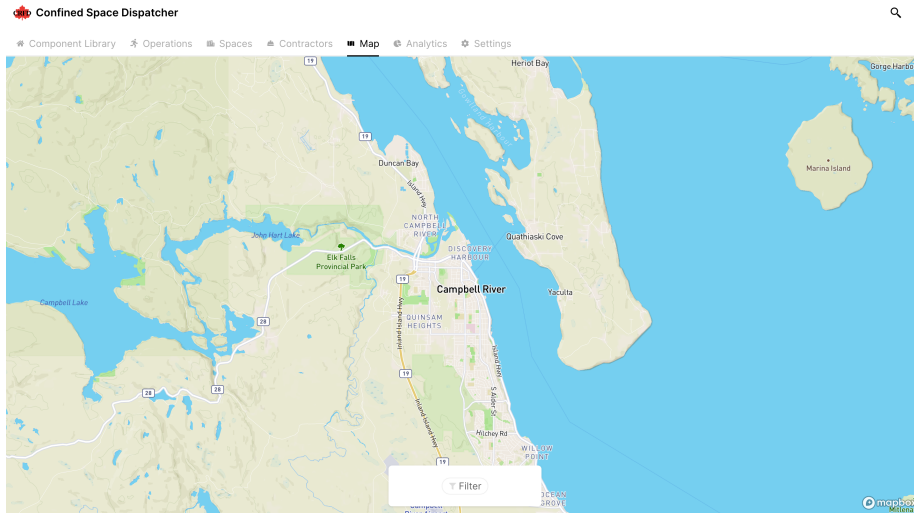


Figure 25: Dispatcher Map Page

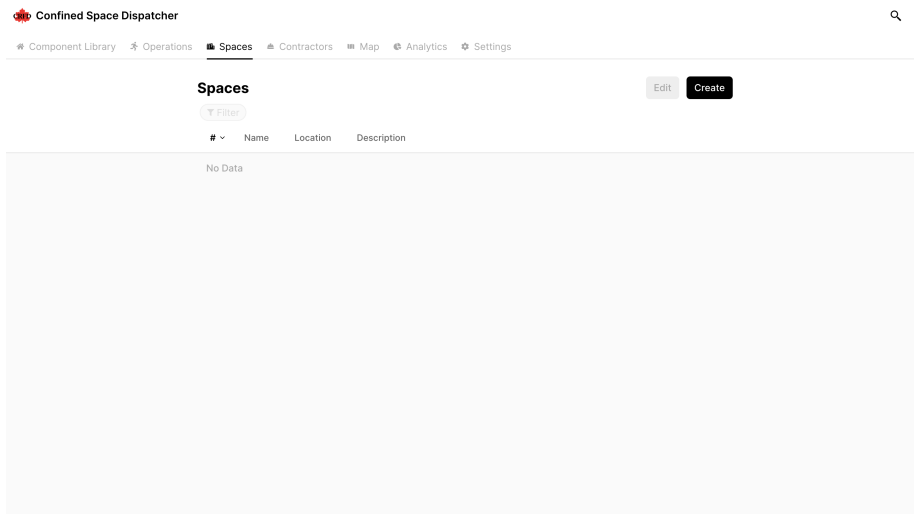


Figure 26: Dispatcher Spaces Table

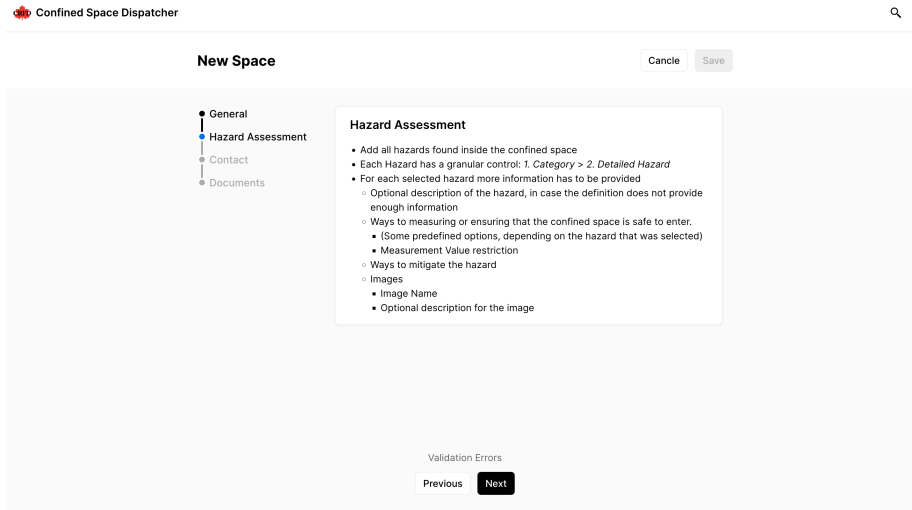


Figure 27: Dispatcher New Space Page

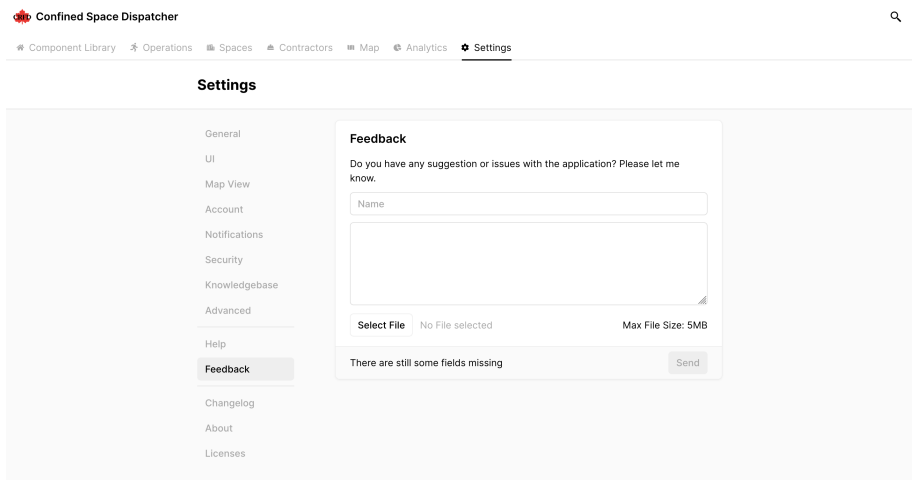


Figure 28: Dispatcher Settings Page

6.3.6 cs-worker

The earlier prototype I have made of this application was made using Swift and SwiftUI. It already possessed the ability to send accelerometer, battery and location data back to the Firebase Firestore database. When the transmission was active the *Terminal* would indicate the active confined space operation. The user would be presented with a misuse warning whenever he would start his work session.

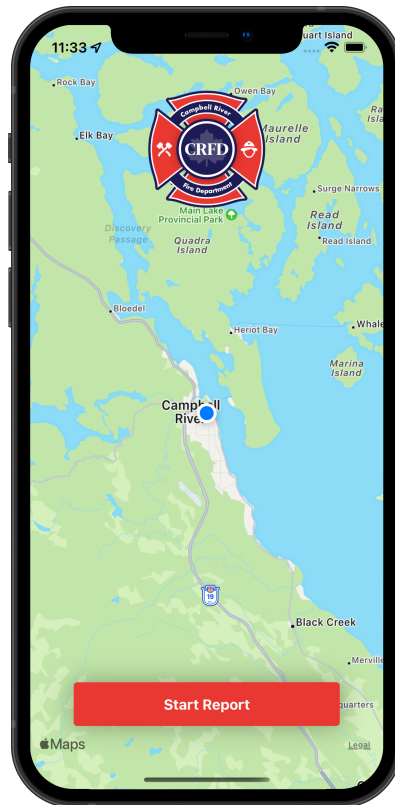


Figure 29: Worker Application Prototype

7 Conclusion

With this thesis I have managed to get more insight into the complete development cycle of a software system. I have learned a lot about researching for a particular topic and how I can gain knowledge to realize a solution to a complex problem. The application should drastically improve the workflow of the fire department as is. Because the fire department chief just left the office it remains to be seen if the development will be continued. In a future meeting I will help the fire department to install the application and discuss further maintenance issues with them.

In the future the application could be tested and connected with on-sight sensors as an interface for this kind of feature is already planned and implemented in the data model.

In this kind of software lies a great potential to further improve safety and security of both workers and confined spaces rescuers. That is why more and more services are being developed and projects with even greater autonomy are being planned and tested. In an E-Mail by the Engineer Luca Pohlmann of the German based Drägerwerk company (a company specializing in the manufacturing of medical and protection equipment) he told me that they are heavily researching in the area of automated inspection in confined spaces through the use of drones and robots.



Figure 30: Spot - Source: Team CoSTAR [70], [71]

8 Note of Thanks

To cap this all of I want to give a huge thanks to:

- **Kelly Bellefleuer** and the entire **Fire Department of Campbell River**. All of them have taken lots of time out of their schedule to make this software a reality, answered all of my questions and given me essential feedback on how to improve the software.
- My professor, **Prof. Dr. rer. nat. Wolfgang Lux** and his assistant **Frederik Feichtmeier B. Eng.** for providing me with help along the way of writing this thesis.
- **Dr. Lisa Spanier** from the HSD ZWEK for her kind support and her advisory on the structure of my thesis.

References

- [1] Inc. Autodesk, “CAD software.” www.autodesk.ca. Accessed: May 15, 2022. [Online]. Available at: <https://www.autodesk.ca/en/solutions/cad-software>
- [2] Wikipedia Contributors, “Computer-aided dispatch.” Wikipedia, Jan. 2021. Accessed: May 15, 2022. [Online]. Available at: https://en.wikipedia.org/wiki/Computer-aided_dispatch
- [3] US Department of Homeland Security, “Tech note - computer aided dispatch systems.” Sep. 2011. Accessed: May 15, 2022. [Online]. Available at: https://www.dhs.gov/sites/default/files/publications/CAD_TN_0911-508.pdf
- [4] APCO International, “Public safety communications common incident types for data exchange.” Dec. 2012.
- [5] Wikipedia Contributors, “9-1-1.” Wikipedia, Jan. 2022. Available at: <https://en.wikipedia.org/wiki/9-1-1>
- [6] ArcGIS by ESRI Inc., “NG9-1-1.” camps-lacounty.hub.arcgis.com. Accessed: May 16, 2022. [Online]. Available at: <https://camps-lacounty.hub.arcgis.com/pages/ng9-1-1>
- [7] US National 911 Program and US National Highway Traffic Safety Administration’s Office of Emergency Medical Services, “911.gov.” www.911.gov. Accessed: May 16, 2022. [Online]. Available at: https://www.911.gov/project_ng911publicsafety/ems/understandingng911.html
- [8] Centralsquare, “Computer-aided dispatch | CAD dispatch software | CentralSquare.” www.centralsquare.com. Accessed: May 18, 2022. [Online]. Available at: <https://www.centralsquare.com/public-safety/cad>
- [9] Kelly Bellefleur, “Images of the campbell river fire department.” Private Message, Jun. 2022.
- [10] City of Campbell River, “Connect campbell river.” Feb. 2019. Available at: <https://www.campbellriver.ca/docs/default-source/business-economy/campbell-river-community-profile.pdf>
- [11] Mapbox, mapbox.com. Accessed: Jun. 30, 2022. [Online]. Available at: <https://www.mapbox.com/maps>
- [12] “Campbell river fire and rescue.” Firefighting Wiki, Mar. 2022. Accessed: May 17, 2022. [Online]. Available at: https://fire.fandom.com/wiki/Campbell_River_Fire_and_Rescue
- [13] Shane MacKichan, “Campbell river fire rescue.” SmugMug, Aug. 2021. Accessed: May 17, 2022. [Online]. Available at: <https://vancouverfirerescue.smugmug.com/Canada/British-Columbia/Strathcona/Campbell-River-Fire-Rescue/i-L9RSWQP>
- [14] Kelly Bellefleur, “In person chat with the deputy chief of the campbell river fire department.” 2022.
- [15] Canadian Department of Justice, “SOR-86-304 canada occupational health and safety regulations,” vol. 11. Feb. 2022. Accessed: Apr. 07, 2022. [Online]. Available at: <https://laws.justice.gc.ca/eng/regulations/SOR-86-304/index.html>

- [16] Government of Canada, “Confined spaces – no easy way out.” Oct. 2021. Accessed: Apr. 07, 2022. [Online]. Available at: <https://www.canada.ca/en/employment-social-development/services/health-safety/reports/confined-spaces.html>
- [17] David McPherson, 3M Canada, and OHS Canada, “Webinar: Confined space safety.” [www.youtube.com](https://www.youtube.com/watch?v=8fd7O8uSy3Q), Jun. 2020. Accessed: May 15, 2022. [Online]. Available at: <https://www.youtube.com/watch?v=8fd7O8uSy3Q>
- [18] WorkSafeBC, “Confined spaces.” Accessed: May 18, 2022. [Online]. Available at: <https://www.worksafebc.com/en/health-safety/hazards-exposures/confined-spaces>
- [19] Drägerwerk AG & Co. KGaA, “Training information confined space entry rescue.” Article, Sep. 2015. Accessed: May 11, 2022. [Online]. Available at: <https://www.draeger.com/Products/Content/confined-space-entry-rescue-9072486-en.pdf>
- [20] Wikipedia Contributors, “British columbia.” Wikipedia, Jul. 2019. Available at: https://en.wikipedia.org/wiki/British_Columbia
- [21] Wikipedia Contributors, “Statutory body.” Wikipedia, Mar. 2022. Accessed: May 18, 2022. [Online]. Available at: https://en.wikipedia.org/wiki/Statutory_body
- [22] Government of British Columbia, “B.c. Reg. 296/97 - occupational health and safety regulation.” Dec. 2021. Accessed: May 18, 2022. [Online]. Available at: https://www.bclaws.gov.bc.ca/civix/document/id/crbc/crbc/296_97_multi
- [23] Wikipedia Contributors, “Lockout–tagout.” Wikipedia, Sep. 2020. Accessed: May 18, 2022. [Online]. Available at: <https://en.wikipedia.org/wiki/Lockout%E2%80%93tagout>
- [24] Foundation of Cannabis Unified Standards Inc., “Standards vs. regulations.” Focus Standards. Accessed: May 16, 2022. [Online]. Available at: <https://www.focusstandards.org/standards-vs-regulations-vs-laws/>
- [25] NFPA -National Fire Protection Association, “NFPA 704: Standard system for the identification of the hazards of materials for emergency response.” [nfpa.org](https://www.nfpa.org), 2022. Accessed: Jun. 26, 2022. [Online]. Available at: <https://www.nfpa.org/codes-and-standards/all-codes-and-standards/list-of-codes-and-standards/detail?code=704>
- [26] Wolframalpha, “Wolfram|alpha: Computational intelligence.” [Wolframalpha.com](https://www.wolframalpha.com), Jun. 2022. Available at: <https://www.wolframalpha.com>
- [27] Michael Mulder and SAP, “NFPA labels and regulatory content update.” blogs.sap.com, Feb. 2022. Accessed: Jun. 26, 2022. [Online]. Available at: <https://blogs.sap.com/2022/02/03/nfpa-labels-and-regulatory-content-update/>

- [28] Roco Rescue Inc., Chris Carlsen, and Tim Robson, “Confined space fatalities...an updated look at the numbers.” blog.rocorescue.com, Sep. 2021. Accessed: May 16, 2022. [Online]. Available at: <https://blog.rocorescue.com/roco-rescue-blog/confined-space-fatalities-an-updated-look-at-the-numbers>
- [29] Nathan Meyer, “Confined space solutions.” Sep. 2019. Accessed: May 17, 2022. [Online]. Available at: <http://www.unitechus.com/wp-content/uploads/2019/09/7Nathan-MeyerFallProtection.pdf>
- [30] Jason Stuart Selmon, “Work-related traumatic fatal injuries involving confined spaces,” PhD thesis, 2019. Accessed: May 17, 2022. [Online]. Available at: <https://espace.curtin.edu.au/bitstream/handle/20.500.11937/79398/Selman%20J%202019.pdf>
- [31] 3M Canada, “Training course: 6880 confined space entry & monitor.” [3mcanada.ca](https://www.3mcanada.ca/3m/en_ca/worker-health-safety-ca/safety-town-square/articles/training-course-6880-confined-space-entry-monitor), May 2019. Available at: https://www.3mcanada.ca/3m/en_ca/worker-health-safety-ca/safety-town-square/articles/training-course-6880-confined-space-entry-monitor
- [32] Worksite Safety, “Confined space online course.” Accessed: May 18, 2022. [Online]. Available at: <https://worksitesafety.ca/product/training/online/confined-spaces/>
- [33] Jason Selman, Jeffrey Spickett, Janis Jansz, and Benjamin Mullins, “Confined space rescue: A proposed procedure to reduce the risks,” *Safety Science*, vol. 113, Mar. 2019, doi: 10.1016/j.ssci.2018.11.017.
- [34] NI911, “Contact page.” Accessed: May 19, 2022. [Online]. Available at: <https://www.ni911.ca/contact-us>
- [35] Statcounter, “Mobile operating system market share canada.” Statcounter Global Stats, Apr. 2022. Accessed: May 19, 2022. [Online]. Available at: <https://gs.statcounter.com/os-market-share/mobile/canada>
- [36] Wikipedia Contributors, “Cross-platform software.” Wikipedia, Nov. 2019. Accessed: May 24, 2022. [Online]. Available at: https://en.wikipedia.org/wiki/Cross-platform_software
- [37] OpenJS Foundation, “Build cross-platform desktop apps with JavaScript, HTML, and CSS.” www.electronjs.org. Accessed: May 24, 2022. [Online]. Available at: <https://www.electronjs.org/>
- [38] Electron and OpenJS Foundation, “Build cross-platform desktop apps with JavaScript, HTML, and CSS.” www.electronjs.org. Accessed: Jun. 27, 2022. [Online]. Available at: <https://www.electronjs.org/>
- [39] Electron and OpenJS Foundation, “Electron apps search=microsoft.” electronjs.org. Accessed: Jun. 27, 2022. [Online]. Available at: <https://www.electronjs.org/apps?q=Microsoft>
- [40] Wikipedia Contributors, “Vue.js.” Wikipedia, Feb. 2020. Accessed: Jul. 08, 2022. [Online]. Available at: <https://en.wikipedia.org/wiki/Vue.js>
- [41] Jeff Delany, “Vue.js explained in 100 seconds.” www.youtube.com, Apr. 2020. Accessed: Jun. 27, 2022. [Online]. Available at: https://www.youtube.com/watch?v=nhBVL41-_Cw

- [42] Evan You and VueJS, “Introduction | vue.js.” vuejs.org. Accessed: Jun. 27, 2022. [Online]. Available at: <https://vuejs.org/guide/introduction.html#the-progressive-framework>
- [43] Wikipedia Contributors, “Firebase.” Wikipedia; Wikimedia Foundation, Jun. 2022. Accessed: Jun. 28, 2022. [Online]. Available at: <https://en.wikipedia.org/wiki/Firebase>
- [44] Google, “Firebase.” Firebase. Accessed: Jun. 28, 2022. [Online]. Available at: <https://firebase.google.com/>
- [45] Todd Kerpelman and Google, “Cloud firestore vs the realtime database: Which one do i use?” The Firebase Blog, Jan. 2019. Accessed: Jun. 29, 2022. [Online]. Available at: <https://firebase.blog/posts/2017/10/cloud-firestore-for-rtdb-developers>
- [46] Mike McDonald and Google, “Firebase - what is the name of the language used for cloud firestore security rules?” Stack Overflow, Oct. 2017. Accessed: Jun. 29, 2022. [Online]. Available at: <https://stackoverflow.com/a/46608881/4179020>
- [47] Google, “Security rules language | firebase documentation.” Firebase, Jun. 2022. Accessed: Jun. 29, 2022. [Online]. Available at: <https://firebase.google.com/docs/rules/rules-language>
- [48] Google, “Common expression language.” GitHub, Jun. 2022. Accessed: Jun. 29, 2022. [Online]. Available at: <https://github.com/google/cel-spec>
- [49] Todd Kerpelman and Rachel Myers, “Unit testing security rules with the firebase emulator suite.” www.youtube.com, Jun. 2020. Accessed: Jun. 20, 2022. [Online]. Available at: <https://www.youtube.com/watch?v=VDulvfBpzZE>
- [50] Google, “Control access with custom claims and security rules | firebase documentation.” Firebase, Jun. 2022. Accessed: Jun. 29, 2022. [Online]. Available at: <https://firebase.google.com/docs/auth/admin/custom-claims>
- [51] Frank van Puffelen and Google, “Server - firebase multi programming language support.” Stack Overflow, Jul. 2020. Accessed: Jun. 30, 2022. [Online]. Available at: <https://stackoverflow.com/a/62760845/4179020>
- [52] Anish Athalye and MIT, “Lecture 6: Version control (git).” missing.csail.mit.edu, Jan. 2020. Accessed: Jun. 30, 2022. [Online]. Available at: <https://missing.csail.mit.edu/2020/version-control/>
- [53] Stack Overflow, “Stack overflow developer survey 2021.” Stack Overflow, May 2021. Available at: <https://insights.stackoverflow.com/survey/2021>
- [54] Git, “Git.” Git-scm.com, 2019. Accessed: Jun. 30, 2022. [Online]. Available at: <https://git-scm.com/>
- [55] Wikipedia Contributors, “Git.” Wikipedia; Wikimedia Foundation, May 2019. Accessed: Jun. 30, 2022. [Online]. Available at: <https://en.wikipedia.org/wiki/Git>
- [56] GitHub, “Build software better, together.” GitHub. Accessed: Jun. 30, 2022. [Online]. Available at: <https://github.com/about>

- [57] Max Rehkopf and Atlassian, “Automated software testing for continuous delivery.” Atlassian. Accessed: Jun. 30, 2022. [Online]. Available at: <https://www.atlassian.com/continuous-delivery/software-testing/automated-testing>
- [58] Paul J. Leach, Microsoft, Rich Salz, Inc. DataPower Technology, Michael H. Mealling, and LLC Refactored Networks, “A universally unique Identifier (UUID) URN namespace.” IETF - Internet Engineering Task Force, Jul. 2005. Accessed: Jun. 24, 2022. [Online]. Available at: <https://datatracker.ietf.org/doc/html/rfc4122#section-1>
- [59] IETF -Internet Engineering Task Force, “About the IETF.” 2019. Accessed: Jun. 23, 2022. [Online]. Available at: <https://www.ietf.org/about/>
- [60] Dr. Julian Onions, “RFC (request for comment) explained - computerphile.” www.youtube.com, Feb. 2021. Accessed: Jun. 23, 2022. [Online]. Available at: <https://www.youtube.com/watch?v=8IXLpoN8Xj0>
- [61] Mark Nottingham, “How to read an RFC.” mnot.net, Jul. 2018. Accessed: Jun. 23, 2022. [Online]. Available at: https://www.mnot.net/blog/2018/07/31/read_rfc
- [62] Emberjs, “RFC stages.” emberjs.github.io. Accessed: Jun. 23, 2022. [Online]. Available at: <https://emberjs.github.io/rfcs/0617-rfc-stages.html>
- [63] Paul C. Bryan, Salesforce, Mark Nottingham, and Akamai, “RFC 6092 - JavaScript object notation (JSON) patch.” IETF - Internet Engineering Task Force, Apr. 2013. Accessed: Jun. 23, 2022. [Online]. Available at: <https://datatracker.ietf.org/doc/html/rfc6902/>
- [64] Dharmafly, “JSON patch.” jsonpatch.com, Jun. 2022. Accessed: Jun. 23, 2022. [Online]. Available at: <https://jsonpatch.com/>
- [65] Microsoft, “Documentation - object types > property modifiers.” www.typescriptlang.org, May 2022. Accessed: Jun. 23, 2022. [Online]. Available at: <https://www.typescriptlang.org/docs/handbook/2/objects.html#property-modifiers>
- [66] Apple Inc., “The basics — the swift programming language (swift 5.7).” docs.swift.org. Accessed: Jun. 23, 2022. [Online]. Available at: <https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html#ID330>
- [67] Jeff Delaney, “Terraform in 100 seconds.” www.youtube.com, Jul. 2021. Accessed: Jul. 01, 2022. [Online]. Available at: <https://www.youtube.com/watch?v=tomUWcQ0P3k>
- [68] Evan You and Vite Contributors, “Vite.” main.vitejs.dev, 2022. Accessed: Jul. 03, 2022. [Online]. Available at: <https://main.vitejs.dev/>
- [69] LLC ActiveCampaign, “Postmark: Application email service with exceptional delivery.” postmarkapp.com. Accessed: Jul. 06, 2022. [Online]. Available at: <https://postmarkapp.com/>
- [70] Evan Ackerman, “Team CoSTAR trains robots for exploring caves on earth and space.” *IEEE Spectrum*, Dec. 2020. Accessed: Jul. 11, 2022. [Online]. Available at: <https://spectrum.ieee.org/team-costar-robots-earth-space>

- [71] Amanda Bouman, Muhammad Fadhil Ginting, Nikhilesh Alatur, Matteo Palieri, David D. Fan, Thomas Touma, Torkom Pailevanian, Sung-Kyun Kim, Kyohei Otsu, Joel Burdick, and Ali-akbar Agha-mohammadi, “Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion,” *arXiv:2010.09259 [cs]*, Nov. 2020, Accessed: Jul. 11, 2022. [Online]. Available at: <https://arxiv.org/abs/2010.09259>

List of Figures

1	Centralsquare CAD - Source: K. Bellefleur [9]	3
2	Map of Canada - Source: Mapbox [11], Author	4
3	2015 Rosenbauer Commander 4000 - Source: S. MacKichan [13]	4
4	Confined Spaces Classification - Source: Canadian Department of Justice [15] (11.01), Author	6
5	Hazardous Confined Space Entry - Source: Drägerwerk AG & Co. KGaA [19]	7
6	Hazard Diamond - Source: NFPA - National Fire Protection Association [25], Author	10
7	Software Release Roadmap	13
8	Feature Mind Map	14
9	Windows Tablet Onboard - Source: K. Bellefleur [9]	15
10	Dispatchment Center Campbell River - Source: NI911 [34]	16
11	Condensed Entity Relationship Diagram	27
12	Contact Entity	29
13	User Entity	30
14	Contractor Entity	30
15	Confined Space Entity	31
16	Operation Entity	33
17	Capture Point Entity	34
18	Kanban Board	36
19	Figma Terminal Indicator States	37
20	Figma Dispatcher Mockup	37
21	Electron Vite Fusion Logo	38
22	GitHub Actions Electron Deploy	38
23	Registration E-Mail	40
24	Vue Router	42
25	Dispatcher Map Page	47
26	Dispatcher Spaces Table	47
27	Dispatcher New Space Page	48
28	Dispatcher Settings Page	48
29	Worker Application Prototype	49
30	Spot - Source: Team CoSTAR [70], [71]	50

List of Tables

1	<i>Dispatcher</i> Component Library	44
---	-----------------------------------------------	----

Glossary

- BC** British Columbia . 9
- CAD** Computer-Aided Dispatch . 2, 13
- CD** Continuous Delivery . 22
- CI** Continuous Integration . 22
- COHSR** Canadian Occupational Health and Safety Regulations . 5, 7, 8, 9
- CSA** Canadian Standards Association . 9
- DevOps** Development Operations . 22
- GUI** Graphical User Interface . 21
- IETF** Internet Engineering Task Force . 28
- NFPA** National Fire Protection Association . 9, 10
- NG9-1-1** Next-Generation 911 . 2, 34, 35
- npm** Node Package Manager . 18
- REALE** Acronym for Australian researchers proposed rescue procedure . 12
- RFC** Request For Comments . 28
- UI** User Interface . 42
- UUID** Universally Unique Identifier . 27



Written in Markdown.

`KuhlTime/hsd-markdown-thesis`

MIT LICENSED.