Graded Unit Report

[redacted]

Contents

Table of Figures

Introduction

This report provides a walkthrough of the process of creating an application for my client, while this introduction will provide prior information that doesn't necessarily fit into any of the following sections.

"Client" refers to a Fire Station Officer at [redacted], whom the application will be developed for. "Site" refers to the [redacted] in [redacted].

Due to the confidential nature of the Emergency Response Plans used in this project, where applicable real copies have been replaced with example copies to provide an accurate representation of the finished product, without using any confidential material in this graded version. Some material is unable to be replicated without use of confidential content, this will be mentioned along with the reason why where applicable.

Project Aims

In this project I am hoping to release a finished and polished product to a real client in a production environment. I am also hoping to enhance my documentation skills, as I primarily focus on implementation rather than documentation.

I also look forward to creating an aesthetically pleasing frontend, as opposed to using Scanner or JOptionPane.

§1 - Interpretation of the Project Assignment Brief

§1.1 Project Brief

The client would like an application to be developed allowing emergency responders to effectively manage Emergency Response Plans. The system should include basic functionality to add, remove, update and view these plans. As the system will be used at emergency incidents the application should be both efficient at runtime and user friendly. Client would not like users to have to waste time at an incident having to figure out how to work the system.

To align with previous systems used by the client, users should be able to navigate to plans by selecting an area from a satellite view of the site. After selecting an area, a more focused map of the selected area will be shown, allowing the user to select a sub-area. Once the lowest level of area has been selected, the user will be able to select a piece of site equipment so that they can view the associated ERP. All the plans are stored as PDF files and should display once selected, with options to display externally and print.

As these plans may change frequently, and there are hundreds of them to be changed, a simple way to swap out old and new diagrams is essential

The client also suggested that being able to view Material Safety Data Sheets for specific equipment within the system would be beneficial although this would be a secondary function of the application.

§1.2 Brief Clarifications (Background Information)

The client also provided numerous supporting materials to help demonstrate how the finish product should look and function. A PowerPoint file at 658 slides was generously provided. This is the "system" that is currently in use. A copy of the PowerPoint file has not been provided as doing so would breach confidentiality and changing the PowerPoint so that it does not would take longer than is reasonable for the benefit provided. Other than a working version of their current system, the PowerPoint also provides useful material such as satellite images, which would help to create a new system accurately mimicking the old one on the front end.

§1.3 Personal Interpretation

My understanding of the project brief is that the client would like a system allowing emergency responders to view Emergency Response Plans (ERPs) at incidents. Runtime speed and ease-of-use are mission-critical for the system to be successful. A simple way to update plans is essential due to the extensive number of plans the application should hold – having to manually search a large list will **not** suffice.

To view plans, users should be able to navigate a satellite view of the [redacted] site, when a user clicks on an area of the site, the satellite view will zoom into the selected area. The lowest level will be the level at which users can view ERPs for site equipment. Based on clarifications described in §1.2, most of the design should be based on their current system – with backend

§2 Functional and Non-Functional Requirements

FR1	Manage ERPs
FR1.1	Add ERPs
FR1.2	Remove ERPs
FR1.3	Update (swap out) ERPs
FR1.4	View ERPs
FR2	Navigation via satellite view
FR2.1	Enhanced view once an area has been selected
FR2.2	Select site equipment at the lowest level, which should show the associated ERP
FR3	Print ERP on request
FR4	Display externally (open in default application e.g. Acrobat) on request

---

NF1	Loading the application should not cause a significant delay to response of an emergency incident
NF2	Users should be able to use the application without specific training (not including training of the previous system)
NF3	Simple method for updating ERPs with new versions

§3 Potential Solutions

§3.1 Solution 1

The system could be developed using Java and incorporating a Swing front-end and database for PDF storage. The frontend would be simplistic and in a similar style to the current system, as to not overcomplicate the system (fulfilling non-functional requirement NF2). A menu bar with the satellite view of the site underneath, with the various options the system will include. An embedded database will be used to manage the ERPs, with two tables – one holding the location the ERP applies to and another holding the file path for the ERP itself, these will be constrained with a 1-1 relationship. In theory this will provide the support to fulfil non-functional requirement NF1.

Using a Java/Swing solution would reduce implementation time as I already have the technical knowledge to implement it, and it has an easy-to-use plugin[i] for Eclipse (my preferred IDE). The Swing library is older, but because of this includes more features and is a tried-and-tested solution. Extra time would still have to be allocated to research implementation of the database and other necessary libraries, for example

§3.2 Solution 2

The second solution will feature the same front-end design decisions as it meet the requirements perfectly. This solution would also include the same embedded database system. The only difference would be replacing Swing with JavaFX [1]. The JavaFX library is newer, but also implements less elements. Where it excels is the upgraded features that it has implemented – FX includes better event handling, supports properties and can be reskinned with CSS. It also supports touchscreen systems and web integration, which **may** prove beneficial in future iterations.

This solution would give the finished product a more modern frontend with some nice extra features but would require extra time for me to learn the library in enough depth to use it effectively. Due to its newer nature it also includes less unofficial documentation which could make the process more difficult.

§3.3 Solution 3

The third and final potential solution is a web-based application with an external database. This would allow more customisability in the design compared to either java library (more applicable to Swing, as FX does support CSS customisation). It would also improve the ease of adding features like the indicated Material Safety Data Sheets which could simply be added to a new page, rather than having to amend a desktop application. The major drawback in this solution is the necessity of having an internet connection. While the client indicated that internet access is available site-wide, it is not uncommon for issues to occur with any sort of connection which may lead to inability to access to application. In the event network access is lost, non-functional requirement NF1 would fail. The same could be said for Solutions one and two, but a network connectivity issue is more common than hardware failure. This solution also allows for rapid deployment of fixes/changes, whereas a desktop application would have to be manually updated, a website would only need to be updated once externally. This solution also includes a major drawback, due to the confidential nature of the Emergency Response Plans access to the website would need to be controlled – with a desktop application it is much easier to do this. This solution also incurs extra costs for web and database hosting.

[i] https://www.eclipse.org/windowbuilder/

§3.4 Chosen Solution

Comparing all three solutions, solution one seems the most beneficial choice. The ease of implementation for the library, extensive documentation and fact that it fulfils the requirements outweighs the negative aspect of being older and effectively deprecated (although still widely used). While solution 2 provides benefits over the negatives in solution 1, it is difficult to justify the extra time required to learn this library. Solution 3 provides extensive benefits but also a major drawback in the necessity of a network connection and added risk of security breaches.

In this process I will be using the Unified Process. This choice allows iterative and incremental development, while also promoting a lot of risk analysis - testing and evaluation and all iterations [2].

§3.5 Required Resources

Apache Derby will be used for the embedded database. This resource was chosen as it is easily embedded into Java applications and has a small footprint (approx. 3.5MB). To use this software a copy of the license must be included with the completed system. This, along with Apache Derby can be downloaded from their website[i]

For this project I will be using Eclipse as my preferred IDE – it provides features that will speed up the implementation phase of the project. Eclipse IDE can be downloaded from the Eclipse Foundation[ii]

To handle the ERPs in PDF format, I will be using IcePDF, available from Icesoft Technologies[iii].

This project will also require the Java Runtime Environment and Java Development Kit. Both can be downloaded from Oracle's download page under the Java header[iv].

The hardware I will be using is my personal computer, with specifications as follows:

CPU	Ryzen 5 3600
GPU	NVidia GTX 1060 6GB
RAM	16GB Corsair Vengeance
Storage	256GB SSD

This setup also includes dual 27" monitors, and appropriate ergonomic keyboard and mouse.

§3.5 Information Sources

As with any software project, Stack Overflow[v] will undoubtedly prove to be both helpful and completely unhelpful, and it is likely I will use it frequently throughout the implementation phase. It includes a vast library of already answered questions, as well as giving the ability to ask my own development related questions should the need arise.

As the chosen solution uses a Java implementation, Oracle's Java 8 API[vi] will be used to assist development. This documentation also includes details on the Swing toolkit. As this is published by the Java developers, it is the most reliable source of information on the language.

---

[i] https://db.apache.org/derby/
[ii] https://www.eclipse.org/
[iii] http://www.icesoft.org/java/home.jsf
[iv] https://www.oracle.com/technetwork/java/
[v] https://stackoverflow.com/questions
[vi] https://docs.oracle.com/javase/8/docs/api/

For information on the Apache Derby embedded database aspect, I will use the official documentation[i]

The Java PDF library, IcePDF, also includes official documentation[ii]. On the documentation page, the relevant documentation for my version is the JavaDoc API (Viewer) and the Developer's Guide

i http://db.apache.org/derby/docs/10.6/getstart/index.html

ii http://www.icesoft.org/java/projects/ICEpdf/documentation.jsf

§4 Use Case Diagram

§4.1 Initial Use Case Diagram

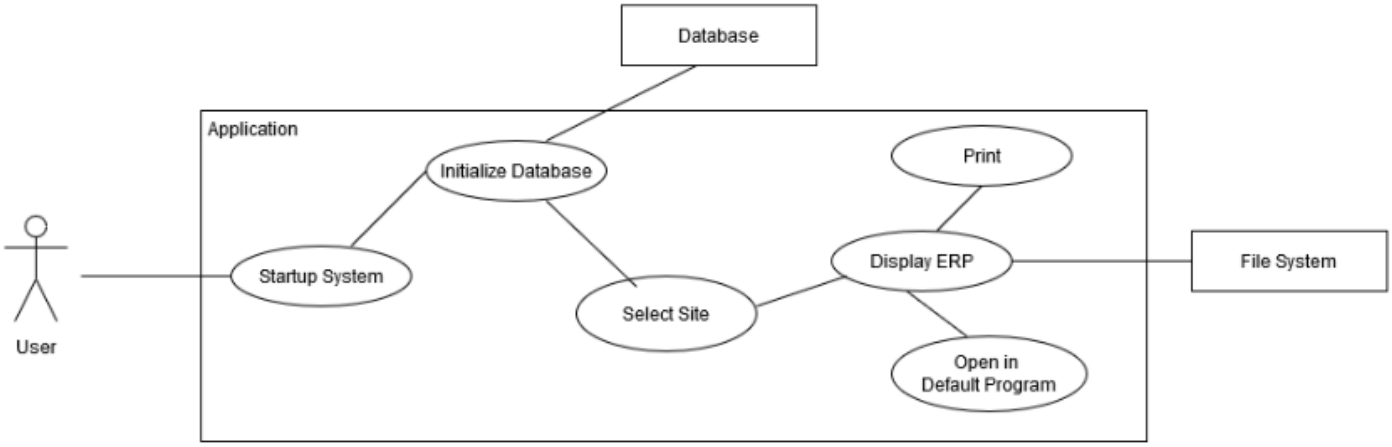


Figure 1 Top Level Use Case

The initial use case diagram shows the barebones functionality of the system. After confirming certain details with the Client, and further analysing the brief the final diagram in *§8 Full Use Case Diagram* was created.

§5 Natural Language Analysis

Green Text represents nouns (entities), while pink text represents verbs (actions). For future reference, yellow text represents non-functional requirements and miscellaneous notables.

The client would like an application to be developed allowing emergency responders to effectively manage Emergency Response Plans. The system should include basic functionality to add, remove, update and view these plans. As the system will be used at emergency incidents the application should be both efficient at runtime and user friendly. Client would not like users to have to waste time at an incident having to figure out how to work the system.

To align with previous systems used by the client, users should be able to navigate to plans by selecting an area from a satellite view of the site. After selecting an area, a more focused map of the selected area will be shown, allowing the user to select a sub-area. Once the lowest level of area has been selected, the user will be able to select a piece of site equipment so that they can view the associated ERP. All the plans are stored as PDF files and should display once selected, with options to display externally and print.

As these plans may change frequently, and there are hundreds of them to be changed, a simple way to swap out old and new diagrams is essential. The client also suggested that being able to view Material Safety Data Sheets for specific equipment within the system would be beneficial although this would be a secondary function of the application.

Based on analysis of the initial brief and after some informal clarifications from the Client, we can deduce the following:

Entities that need to be represented in the system as classes are at the least the Emergency Response Plans (ERPs) containing the reference to the PDF file and the area of the site which will be represented with a Site class. After Client discussion, it was decided that Material Safety Data Sheets should be kept separate from the ERP management system to avoid confusion

Behaviour of the system should include functionality to navigate the site via satellite view, and to select ERPs by navigating to their respective incident locations. To improve on the previous system used by the client, updating outdated ERPs should be simplified as much as possible.

The main constraint of the system based on the brief is that all the plans are stored as PDF documents – this means a specialised library must be used to handle them. Based on client meetings, it was also discovered that the system may be used inside of a roaming vehicle and although a maintained internet connection is there, there is the possibility of connection being lost. This means fail-safes must be put in place to prevent damage to the system and if possible continued use when an internet connection is not established. The Client also indicated that the application would be hosted on a central server and accessed via file sharing, rather than distributed to each machine.

§6 Identification of Analysis Techniques

The chosen design models and respective justifications are described below, as well as a brief summary of why some design models were decided against.

Entity Relationship Diagrams – Although the database for this application is a very simple system, it is important to properly model any database system before commencing with implementation. This ensures a good foundation for maintenance and usage of the database.

Use-Case Diagram – Despite the basic nature of the use case in this system, this diagram takes only minutes to create and may provide a good way of making sure the system is accurately created as intended. It is also helpful to describe the functionality of the system to the client, as it is easy to understand with no complex notation.

Sequence Diagram – At application start up, the database must be initialised. This procedure may be complex, and a sequence diagram should provide the necessary clarification to help both myself and future developers understand how it works. Further sequence diagrams may be created well into the construction phase if any unforeseen complexity arises that would benefit from a diagram.

Colour Charts/Digital Prototype/Wireframe – All 3 of these will be used to provide the client with a preview of how the finished product will look, with the colour scheme demonstrating some of the colours that could be used and allowing the client to select between them. The wireframe will also give the client an idea of the placement of objects within the UI, while the digital prototype will combine both and add basic functionality to provide a detailed understanding of how the finished application will look and feel – in my opinion, this is the best way forward to provide the client with a front-end they are happy with.

Site-map – A site map will be produced to show the hierarchy of all sites and sub-sites. This will be produced during the implementation phase, and will be based off the previous system and a client meeting.

Class Diagram – A class diagram will be produced at the tail-end of development to help build good user documentation. It will model the entire system, and as it is difficult to predict exactly what classes and methods will be required before implementation has begun it will have to be held off until later. Multiple iterations may be made to follow the development lifecycle.

Activity Chart – An activity chart will be produced to provide modelling of loading an ERP from the database. To ensure that it accurately models the system, it will be created alongside the implementation.

State Diagram – I do not feel a state diagram would be justified given the simplicity of the application, and therefore one will not be created as it would not provide any benefit.

§7 Top-Level Entity Relationship Diagram

The database for this application is very simple, and this is reflected in the ERD.

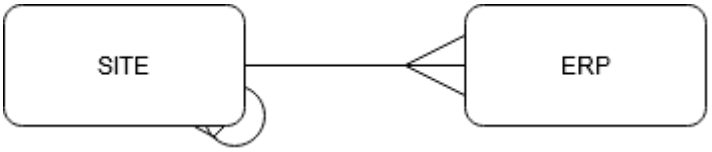


Figure 2 Initial Top-Level ERD

The SITE entity represents both the [redacted] site itself, and any subsites that will be shown in the system. This is achieved by using a one-to-many self-referencing relationship.

The ERP entity represents the actual PDF file. Each Site may have many ERPs, and this is shown in the top level ERD.

§7.1 Amendments

After further consideration, it was decided that separating the PDF BLOBs into their own entity and joining it to the ERP entity via a one to one relationship would improve efficiency. The amended diagram demonstrates this. Optionality was also fixed, with SITE no longer having a mandatory relationship to ERP.

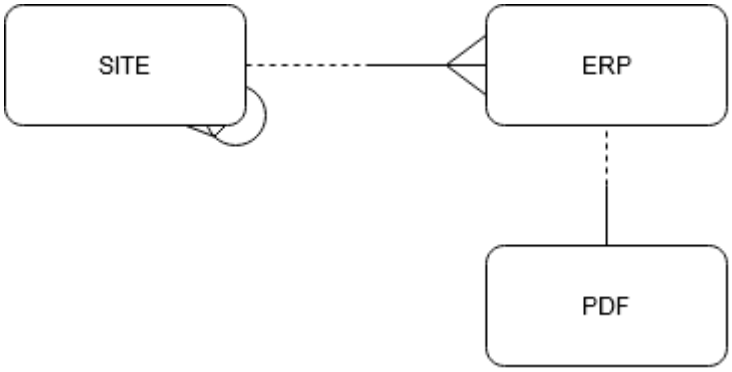


Figure 3 Amended Top Level ERD

§7.2 Final ERD

The completed ERD shows field names, optionality and relationship properties as well as transferability and cardinality.

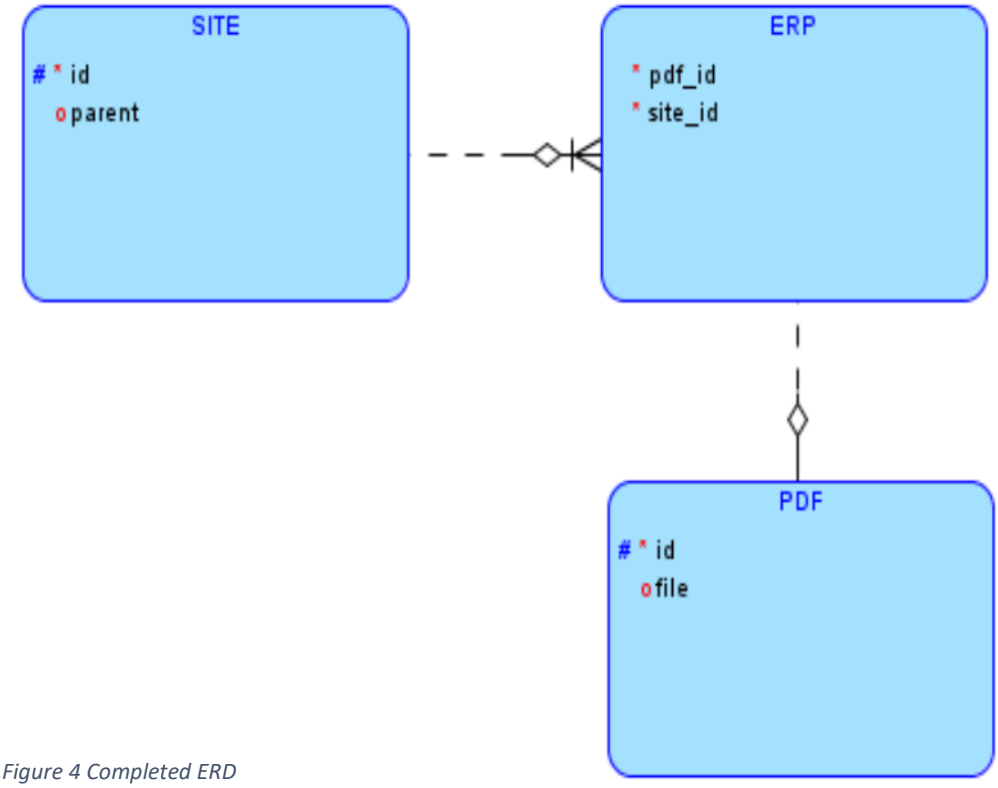


Figure 4 Completed ERD

§8 Full Use Case Diagram

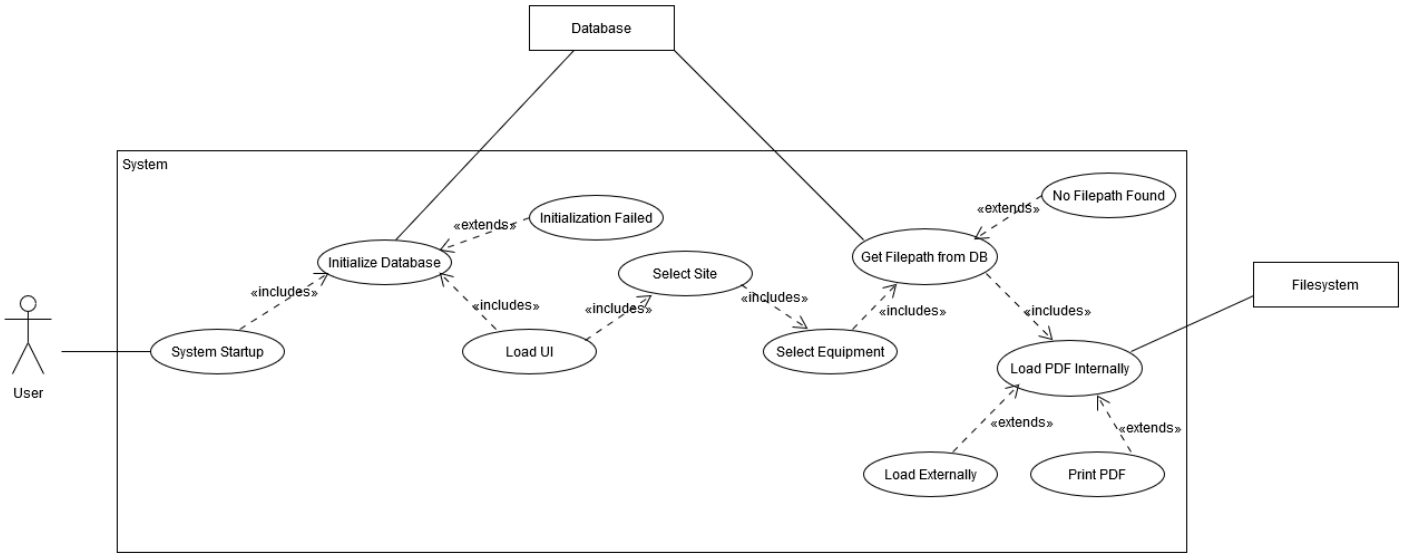


Figure 5 Full UCD

The full Use Case Diagram shows the use case for the completed system after discussions with the client.

§9 Database Sequence Diagram

The sequence diagram shows the procedure for initialising the database and if necessary, creating the tables. It shows how the `DBManager` Object is instantiated, which then instantiates the `Database` object – establishing a connection or returning an error. Any exceptions are thrown back to the driver class, except an `SQLException` with an `SQLState` of 'X0Y32' – this means the table already exists and won't be recreated and is normal.

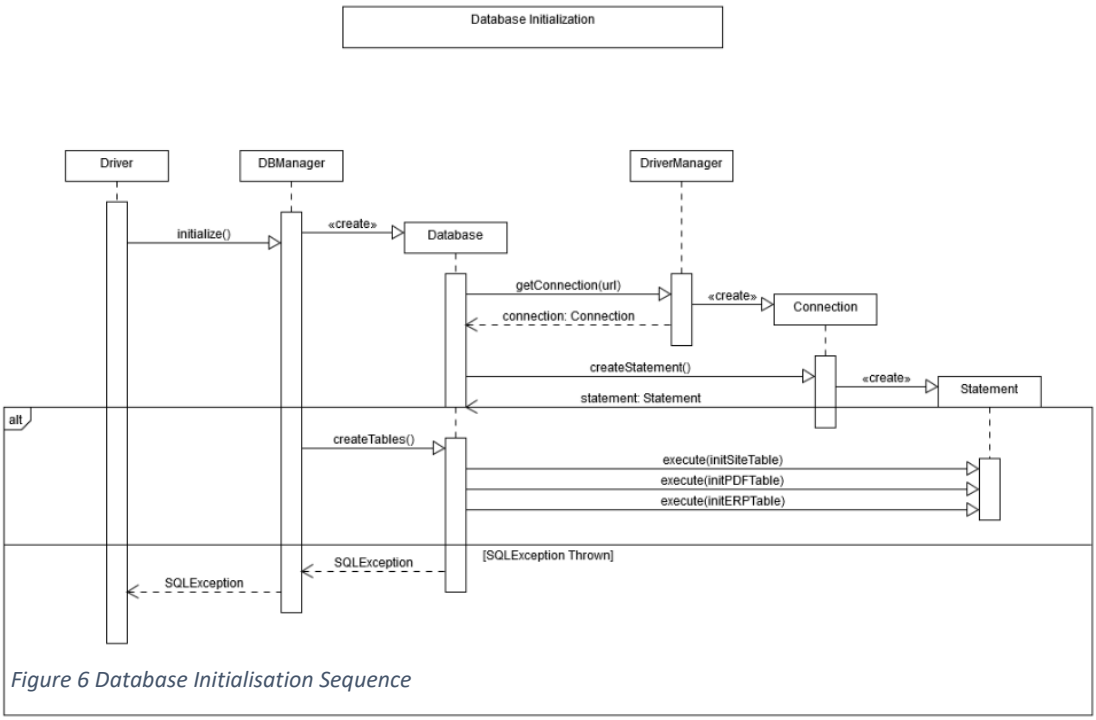


Figure 6 Database Initialisation Sequence

§10 HCI Design Choices

§10.1 Colour Scheme

As this application is not intended for public use and will only be used at emergency incidents, how appealing the colours are is very low on the priority list. Regardless, the colour scheme should not provide a hindrance to the normal use of the program. A basic colour scheme consisting of grey background with light coloured buttons on the sidebar. The satellite view is taken directly from Google Earth so any colours on it are uncontrollable, hence buttons laid over it must be a colour which will appear visible on all backgrounds or at the least backgrounds uncommon on satellite images. For this, we will be using a light blue button with black text – this should provide enough contrast over the typically green/brown background from Google Earth.

§10.2 Font

While the application won't be commercially available it is still important to ensure readability. The font used will be based on the previous system used by the client, which was Calibri, typically sized 24 (smaller or larger sizes where applicable, such as headings and small text fields)

§10.3 Legal

As this application is intended to be used at a specific workplace, failure to supply support for disabled users could qualify as a breach of the Equality Act 2010. On the other hand, it is intended to be used by Firefighters who are subject to strict eyesight and physical requirements. As such, no users will suffer from major eyesight ability in either colour or acuity [3], and none will suffer from any debilitating physical injuries [4] that would require reasonable adjustments to the application.

§11 Front-end Wireframe

A wireframe was constructed to demonstrate the layout of the frontend. It shows pure placement without any colour or functionality and is good for giving an indication of how easily useable the layout of buttons, text, panels etc. is.

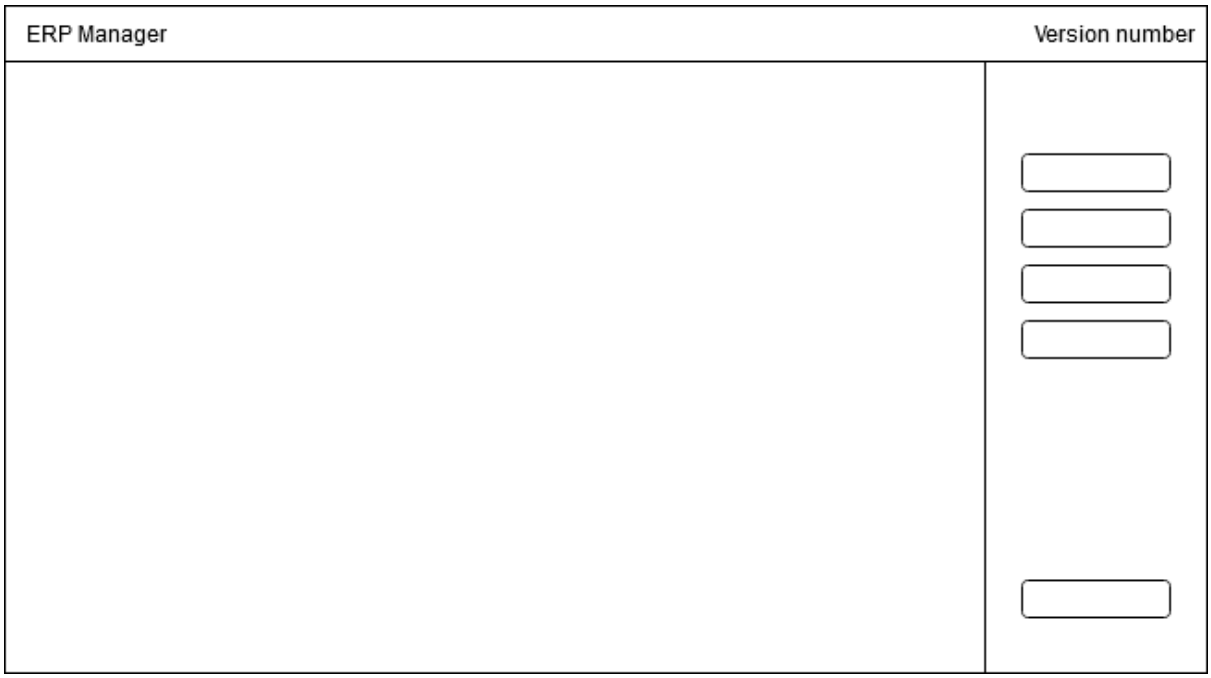


Figure 7 Wireframe Design

§12 Functional Prototype

A prototype was created to showcase basic design and functionality to the client. This can be accessed at [redacted]. Some buttons work and some don't, but it provides a good view of how the finished application will look and feel.

§13 Client Update

On Friday 14 January, a client meeting was had to confirm the design details and showcase the digital prototype. Both the main client, [redacted], as well as the guy in charge[10] of producing the Emergency Response Plans, were present to provide insight. Both were happy with all design choices and offered no alternatives or changes and was overall pleased with the progress and predicted outcome. Other notable points of discussion are below:

1. Client decided that inclusion of Material Safety Data Sheets should be disregarded, as it may further complicate the system unnecessarily.
2. Client queried as to how users would be able to add additional sites rather than just ERPs. This was overlooked in my analysis and no plan was in place for this, as it was assumed that no major site development would be taking place during the lifetime of the application. Regardless of whether this feature had been discussed earlier, the implementation time would take it outside the scope of this project and was decided against.
3. The client mentioned how this application will be replaced by another work-in-progress system in 5 years' time. The usage lifetime of the application can be decreased to accommodate this.
4. Client reiterated the confidentiality of the ERPs, and we discussed how I would be protecting them. Client was happy with proposed protections.
5. Client mentioned that future iterations of ERPs will have equipment lists embedded into the PDF file, so separate functionality that was demonstrated in the prototype is not necessary. It was deemed worthless to create a second prototype for what is essentially removing 1 button, so one was not created.

[10] I have forgotten his name and job title, apologies.

§14 Fully Dressed Use Cases

§14.1 UC1 – Loading the Application

Pre-conditions: Java Runtime Environment must be installed
Post-conditions: Application will be loaded, including the database connection.

Normal Flow:

1. User will launch the application from the Operating System
2. Application will begin loading
3. `Main()` method will create an instance of the `DBManager` class
4. The constructor for the `DBManager` class will attempt to create an instance of the `Database` class
5. The `Database` constructor will attempt to create a `Connection` with the default database URL
6. The `Database` constructor will create a statement object for querying the database
7. The `DBManager` class will call the `createTables()` method in the newly instantiated `Database` object. This will create the database tables if they do not already exist.
8. The `Driver` class will store the instantiated `DBManager` object

Exceptional Flows:

2. No Java Runtime Environment is present, or the wrong version is installed
2.1. The application will not load, an unknow
5. The connection could not be established, therefore no `Connection` object was created
5.2. The `Database` constructor will throw an `SQLException`, aborting the creation of this object.
5.3. The `DBManager` constructor will throw the same `SQLException`, aborting the creation of this object.
5.4. The `Driver` class will stop any further loading and inform the user that an internal error has occurred.
5.5 The use case will terminate at this step; the program will not load.

§14.2 UC2– Viewing an ERP

Pre-conditions: Database connection must be established
Post-conditions: An ERP PDF file will be loaded from the database.

Normal Flow:

1. The user will select a top-level site from the navigation view
2. The user will select a sub site, and any further sub sites within that
3. Once at the lowest level, the user will select a piece of site equipment
4. The ERP will be retrieved based on the Site ID and ERP ID at this location
5. The ERP will be loaded into the Swing UI

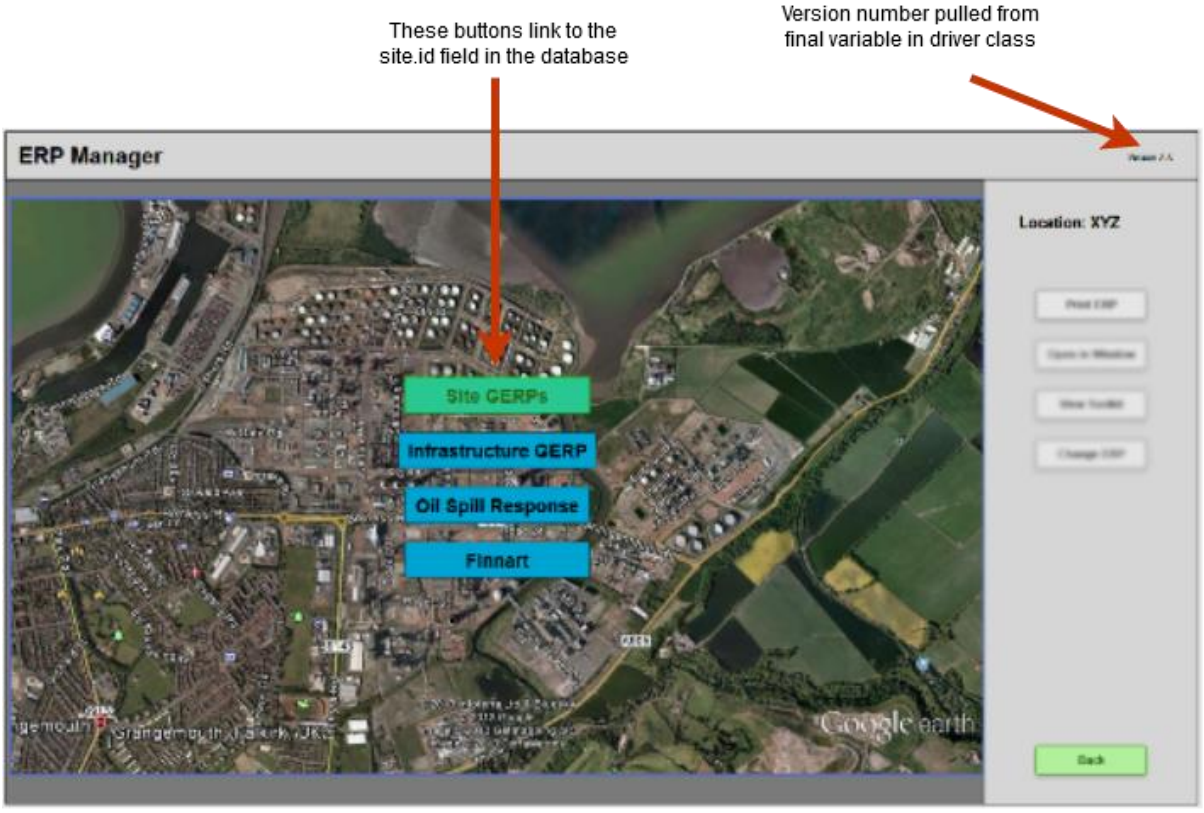
Extension Flows:

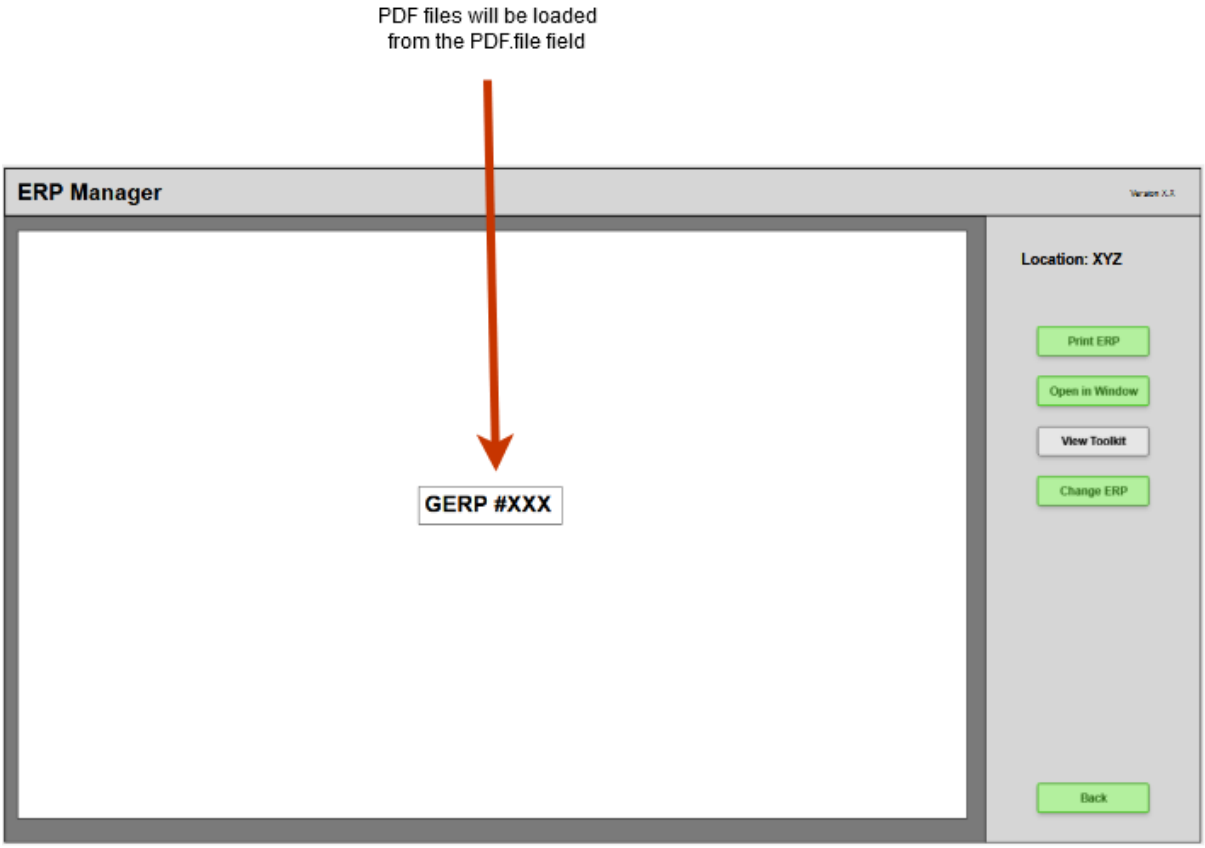
4. No ERP is available for the given site
4.1 An error message will be shown, giving the user the option to add an ERP or to abort

4.2 The user selects to add a new ERP
4.2.1 A windows file explorer window will open, allowing the user to select a file
4.2.2 The selected file will be uploaded to the database. The use case will terminate at this step, and the user will be return to the main menu

5.3 The user aborts
5.3.1 The user will be returned to the main menu, no new ERP will be added. The use case will terminate at this step

§15 Data Binding Model





References

[1] D. Lowe, “10 Differences Between JavaFX and Swing,” [Online]. Available: https://www.dummies.com/programming/java/10-differences-between-javafx-and-swing/. [Accessed 25 January 2020].

[2] “Unified Process,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Unified_Process. [Accessed 29 January 2020].

[3] “Eyesight Requirements,” Scottish Fire and Rescue, [Online]. Available: https://www.firescotland.gov.uk/work-with-us/eyesight-requirements.aspx. [Accessed 23 February 2020].

[4] “Fitness Tests,” Scottish Fire and Rescue, [Online]. Available: https://www.firescotland.gov.uk/work-with-us/fitness-tests.aspx. [Accessed 23 February 2020].