

**Delirium in Paediatric ICUs**

Silicon Valley Samurai (SvS)

Date: 09-05-2023

Adam Logan, Sections: [1](#_heading=h.etc0xq1fdkn0), [2.2](#_heading=h.ivck15vhlsnz), [4.3](#_heading=h.i980293xyuc5), [5.1](#_heading=h.4vt9sx3h1xdr), [5.3](#_heading=h.t6snfsmo4nhz); Contribution: 33%

Robbie Duncan, Sections: [2](#_heading=h.34nfbtfxsxfe), [5.3](#_heading=h.t6snfsmo4nhz); Contribution: 20%

Ewan Forsythe, Sections: [4](#_heading=h.yue2svjus2v4), [5.2](#_heading=h.jofc39n04spb), [5.3](#_heading=h.t6snfsmo4nhz); Contribution: 24%

Andrew Robb, Sections: [3](#_heading=h.cx2hhoewetnx), [5.3](#_heading=h.t6snfsmo4nhz); Contribution: 23%

Team Report: Interim Report (SDP2a)

CSC3068, Software Development Practice

Table Of Contents

[**1. Requirements 2**](#_heading=h.etc0xq1fdkn0)

[1.1. Problem Statement 2](#_heading=h.kyexdvwmlkoi)

[1.2. Elicitation & Analysis Process 2](#_heading=h.e79f7fknwt1p)

[**2. Design 5**](#_heading=h.34nfbtfxsxfe)

[2.1. Software Architecture 5](#_heading=h.3pgi75ckd9b4)

[2.2. System Design 5](#_heading=h.ivck15vhlsnz)

[2.3. UI Design 8](#_heading=h.pjw9ru7wfo9v)

[**3. Implementation 9**](#_heading=h.cx2hhoewetnx)

[3.1. Features Implemented 9](#_heading=h.denf3h8v4aea)

[3.1.1. Audit Log In 9](#_heading=h.ssapp3xp19lw)

[3.1.2. Postgres Database 9](#_heading=h.xyhefcbck78i)

[3.1.3. Audit Graph Visualisation 9](#_heading=h.cnz2x5o00dw3)

[3.1.4. Audit Data Input Form 10](#_heading=h.w0sfq4jg8lbq)

[3.2. Technology Components 10](#_heading=h.awrmoiefecdn)

[3.3. Technology Choice Reflection 12](#_heading=h.3kykgvntauza)

[3.4. Forecasted Technology Choices 12](#_heading=h.elcvn3b6ll5y)

[3.4.1. PuTTy for server 12](#_heading=h.bjjuitf5f68x)

[3.4.2. HTTPS 12](#_heading=h.3pskvzisioft)

[3.5. Third-Party Components & Code 12](#_heading=h.5lae9nkcqt7x)

[3.5.1. Use of Bootstrap 12](#_heading=h.kj5o0fai1i1g)

[3.5.2. Use of External Resources 12](#_heading=h.s52uh4253qzk)

[**4. Testing 13**](#_heading=h.yue2svjus2v4)

[4.1. Code Test Plan 13](#_heading=h.hputwr6i1uii)

[4.1.1. Introduction 13](#_heading=h.peoto68zm959)

[4.1.2. Testing Approach 13](#_heading=h.ffw7zoy3w0ep)

[4.1.3. Scope 13](#_heading=h.yuwfjv3sgztn)

[4.2. Code Testing 14](#_heading=h.rjuid7kyy7el)

[4.2.1 Unit Testing 14](#_heading=h.qyfxuiyclj8c)

[4.2.2 Coverage Based Testing 15](#_heading=h.91bs9ikkkmve)

[4.2.3 Code Reviews 15](#_heading=h.jav6v9wxk2e0)

[4.3. User Evaluation 16](#_heading=h.i980293xyuc5)

[**5. Project Management: Roadmap and Sprint Plan Updates 17**](#_heading=h.sc7dswz3cdbp)

[5.1. Project Management Tools 17](#_heading=h.4vt9sx3h1xdr)

[5.1.1. Managing Documentation and Meetings 17](#_heading=h.wxd2wu5hdfcf)

[5.1.2. Managing Development 17](#_heading=h.17yu3q6cmwkx)

[5.1.3. Links for Online Work Environment 17](#_heading=h.trrkkj5bc2c2)

[5.2. Gantt Chart 18](#_heading=h.jofc39n04spb)

[5.3. Sprint Plan 19](#_heading=h.t6snfsmo4nhz)

[5.3.1. Sprint Description 19](#_heading=h.o1xrrz4tsy09)

[5.3.2. Reviewed Risk of Features 20](#_heading=h.o9mhdo9lhbig)

[**6. Appendices 21**](#_heading=h.gseh4aik73y4)

[Appendix 1: Use Case Descriptions 21](#_heading=h.oo36hsbsccxg)

[Appendix 2: Photos 22](#_heading=h.l8wn3nwqzrz5)

[**7. References 25**](#_heading=h.3zjoerw13v4a)

# 1. Requirements

## 1.1. Problem Statement

Delirium, specifically within children, is an often-misunderstood illness by both medical professionals and within the general public. This is a prevalent issue within Paediatric Intensive Care Units (PICUs) across the UK and ROI.

Further research and education need to be carried out, and therefore 2 systems have been proposed to tackle each of these issues. An ‘audit’ system that will facilitate research within this field will provide a platform to record and visualise compliance data within PICUs, regarding delirium.

The other system will focus on the education aspect of this problem. By developing an e-learning package, which can be modified to contain the contemporary knowledge of paediatric delirium. Including interactive content within the package that will increase engagement and therefore increase the likelihood that the information will be retained by the learner.

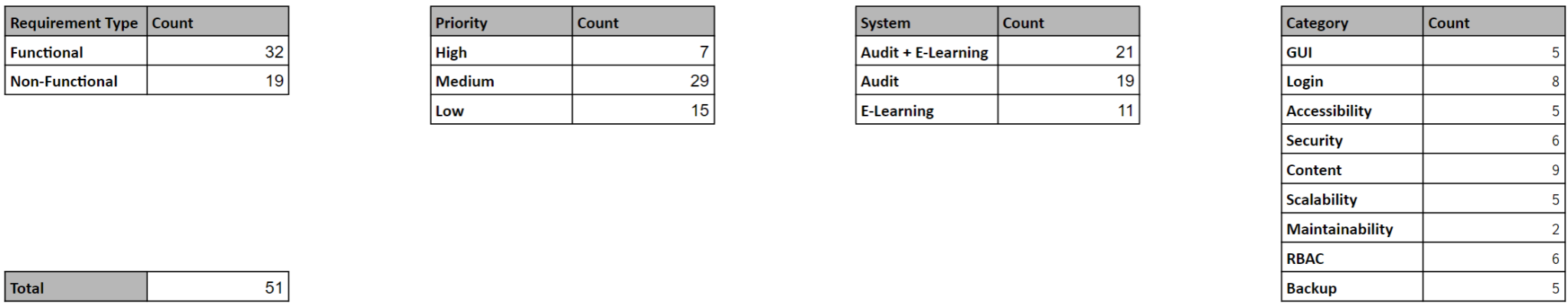
## 1.2. Elicitation & Analysis Process

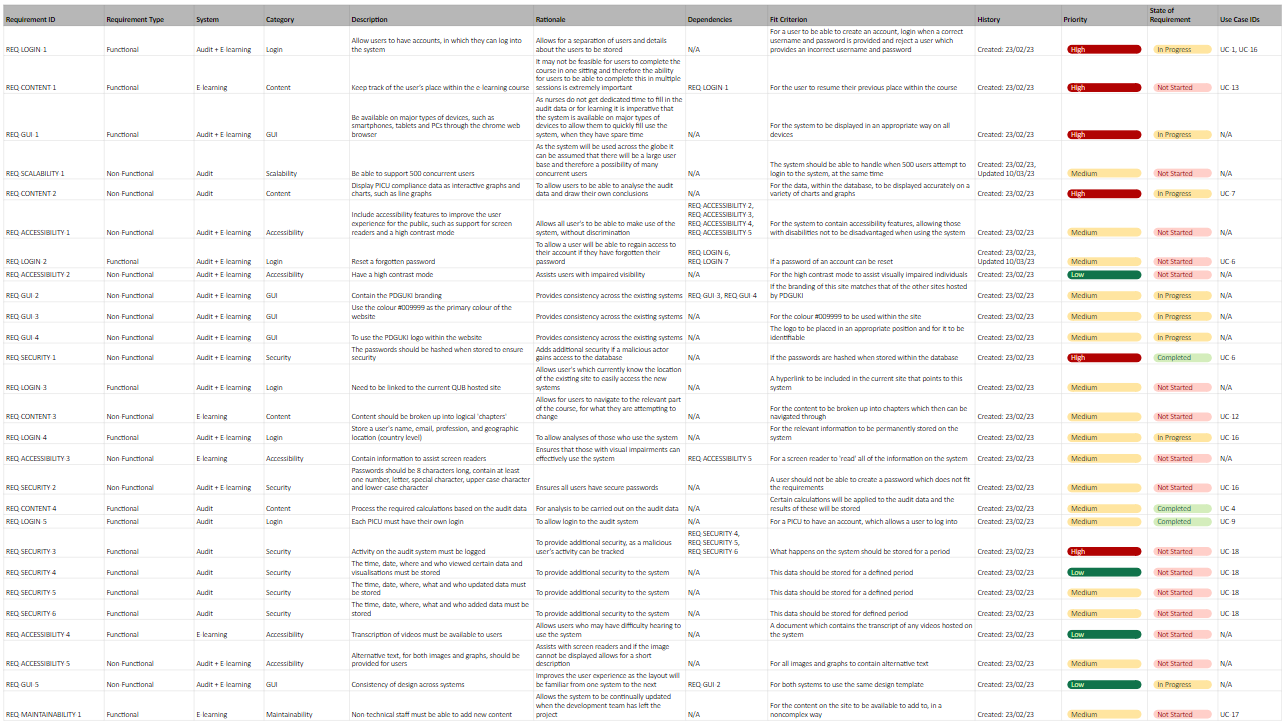
The main method of eliciting the requirements was through the detailed brief provided by Lisa McIlmurray, the main end-user stakeholder. By developing the user stories within section 2 of the first submission, further requirements were produced by analysing the problem more thoroughly. Further along within the elicitation process interviews were conducted between the requirement engineers and the relevant stakeholders, Charles Gillian (Technical Support) and Lisa McIlmurray. The interview questions were derived from the earlier steps within the elicitation process to clarify on points mentioned within the brief.

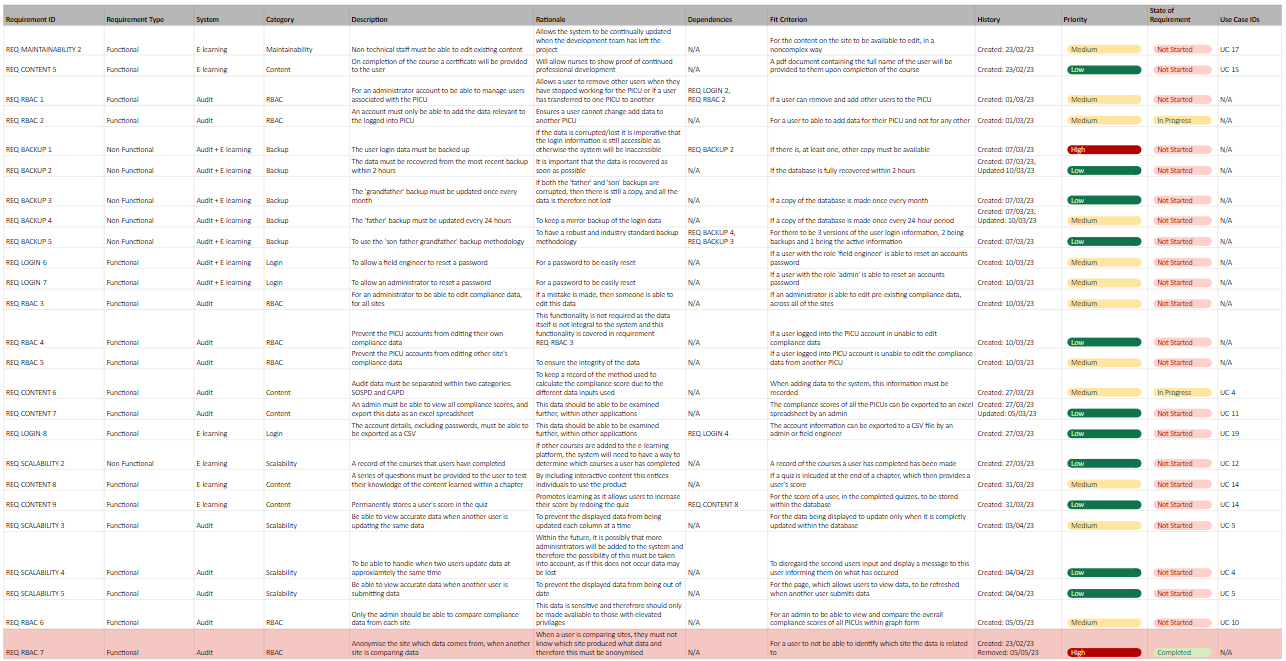
The ‘Structured Natural Language’ notation [[1]](#bookmark=kix.mzg9l6kyh8dj) was used to promote the standardisation of the requirements.

Within analysing the requirements, a prioritisation system was developed to assist the development stage of the project. This consists of three levels: “High”, “Medium” and “Low”. The “High” and “Medium” requirements are essential in delivering to the client. The “Low” requirements are in the “waiting room” [[2]](#bookmark=kix.38wisk44x8ee) which is to say that, depending on time constraints, they may be implemented although not necessary in order for a ‘completed’ product and are candidates for future implementation.

[Figure 1.2.1](#bookmark=id.ks876c5a10w9) displays the number of requirements within each category.

Figure 1.2.1: A summary of the requirements

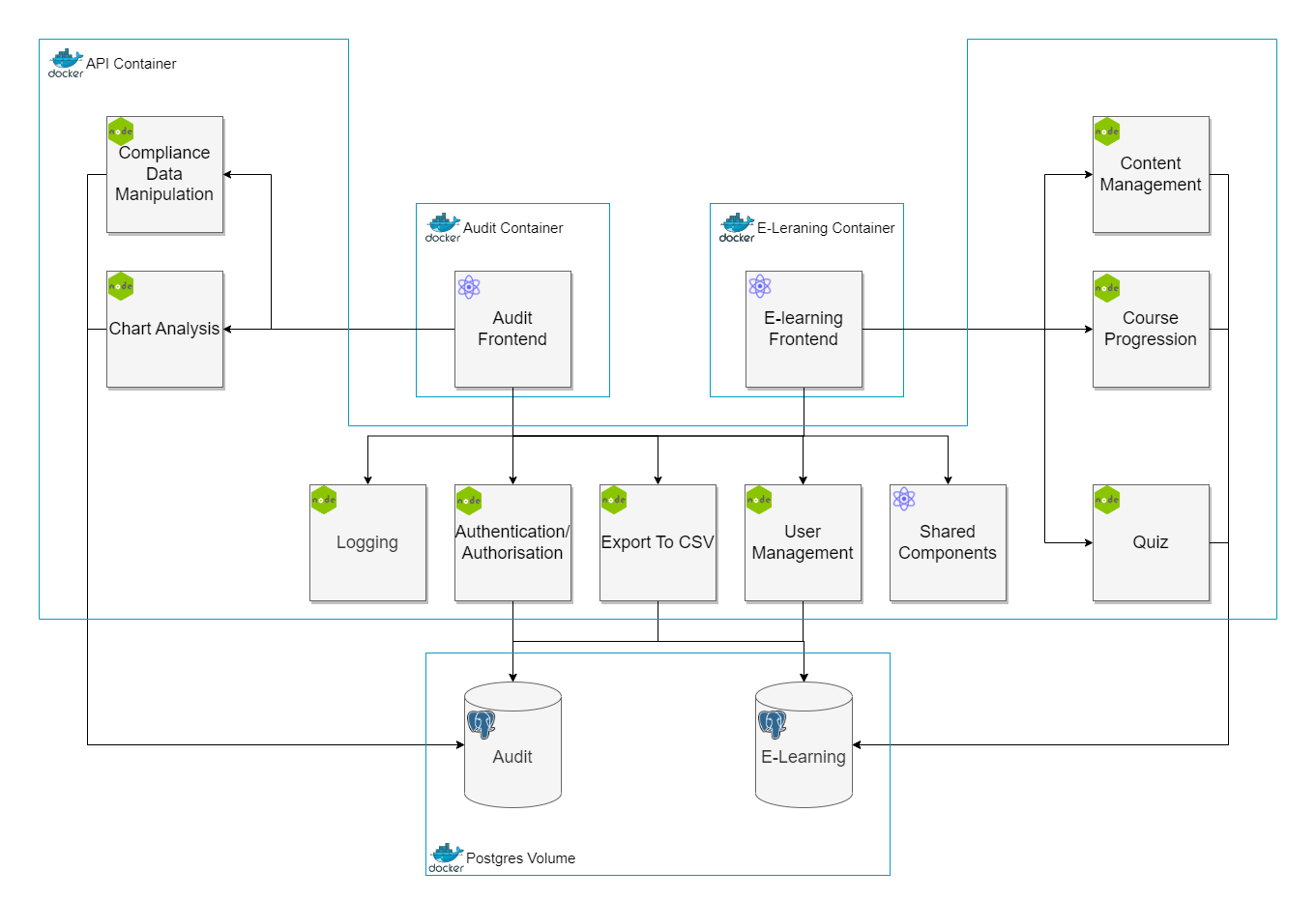
*Figure 1.2.2: The requirements for both systems*



# 2. Design

## 2.1. Software Architecture

During the design phase of the project, the team encountered some issues that needed to be discussed surrounding the architecture of the system. There are multiple systems which will interact and connect in different ways to each other, and the discussions surrounding the connection of these systems using APIs and docker. One of the main points would be whether the Audit and E-learning systems would be separated or kept connected as a single system to more easily reuse common code. It was decided by the team that the Audit and E-learning systems would be separated and use shared components to create the two separate systems along with the APIs ([Figure 2.1.1](#bookmark=id.1jj7n1a2m79k)).

The team decided that a service oriented architecture would be best suited for the project because it would allow for the customisation and adaptation [[3]](#bookmark=id.8l9bnk8gx5u7) of each capability on their own and allow them to be connected, only where they need to be. It also allows for easy and quick creation and debugging of new services which can mean swift capability addition and speedy fixes for existing services. This type of architecture also allowed the team to split up and assign work in a simpler fashion because when each member of the team has their own service to work on, problems which could occur when working on one larger system do not appear as often such as working on functions that conflict or changing pieces which could stop other functions working. 

*Figure 2.1.1: System Architecture Block Diagram*

## 2.2. System Design

For how the system functions, use-case diagrams were created for the Audit and E-learning systems respectively (Figure [2.2.3](#bookmark=id.bjyq8zz2ynov) and [2.2.4](#bookmark=id.f0asu9s20yfs)) to allow the system to be created with a greater understanding of what way it has been planned. These diagrams were created using the understanding of what is needed by the project champion and an understanding of how we would need to put the systems together. The Audit use-case diagram was used to put together the website and system which allows for the 3 different types of users to interact with the system in their own unique way, as per the RBAC [requirements](#_heading=h.e79f7fknwt1p).

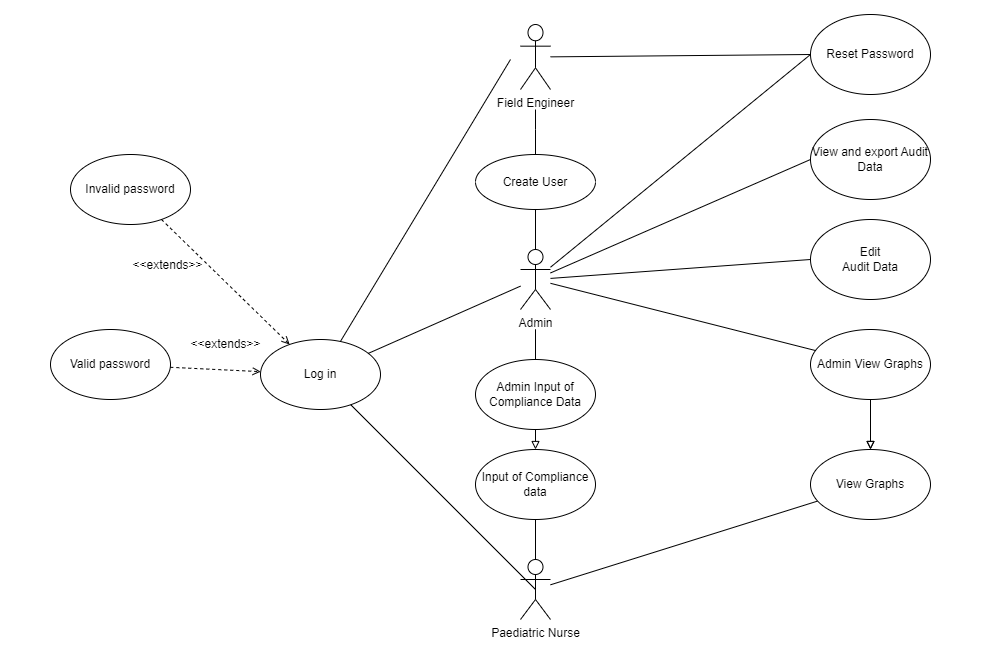
There were discussions based around the database and how it was going to be set up, whether there would be two separate databases, one for the audit system and one for the E-learning, and what tables were going to be created for the database. The actual setup of these databases and the different connections between the tables allow us to have the best functionality and allow us to meet the [requirements](#_heading=h.e79f7fknwt1p) that are relevant to the database which were chosen based on what the project champion was looking for when it came to database functionality. Below is shown the ER diagrams to show how the tables will relate in each database and the Use Case diagrams for each user pathway.

### 

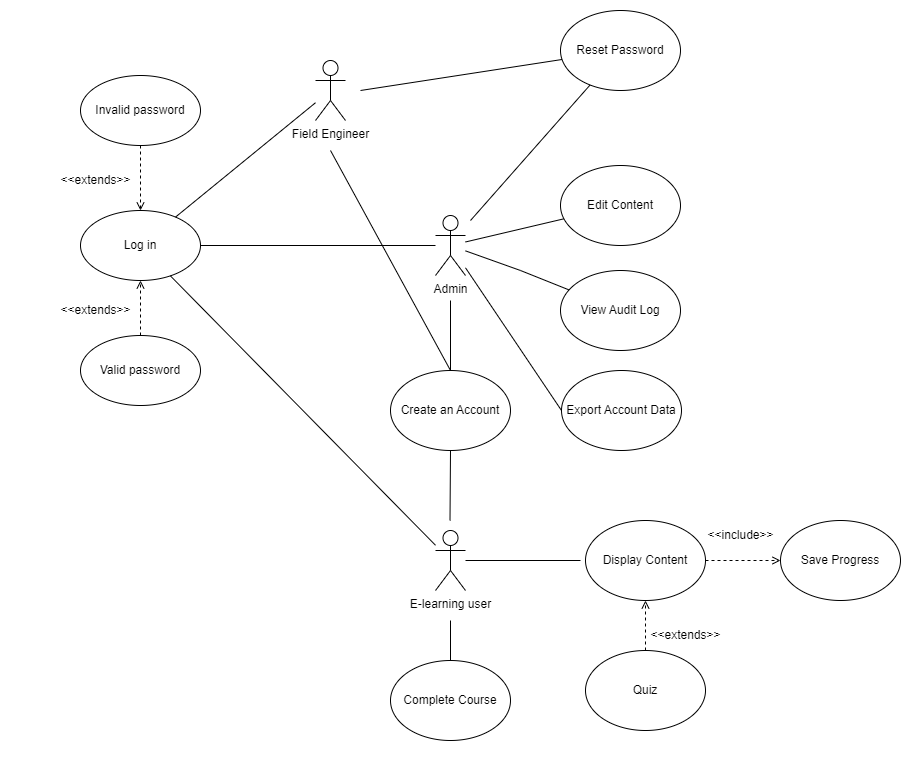
*Figure 2.2.1: The Audit ER Diagram*

### 

*Figure 2.2.2: The E-Learning ER Diagram*



*Figure 2.2.3: The Audit Use Case Diagram (Use case descriptions can be seen in* [*Appendix 1*](#_heading=h.oo36hsbsccxg)*)*



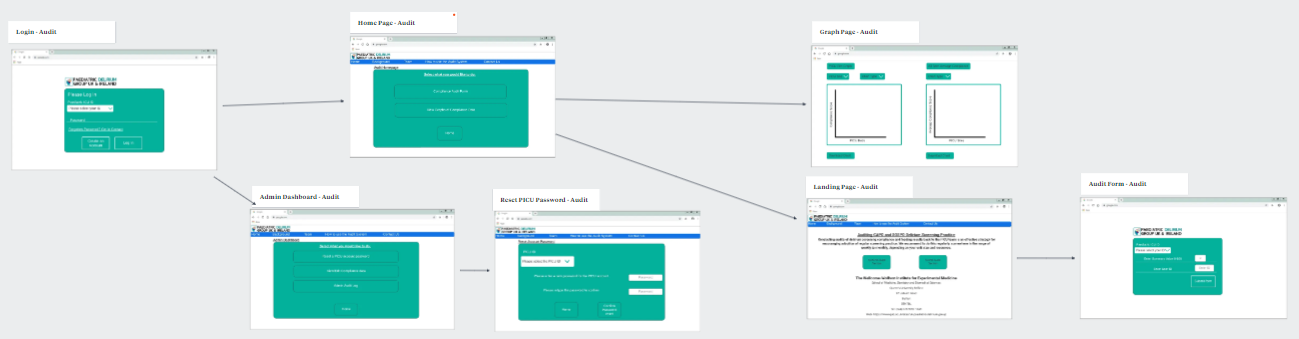
*Figure 2.2.4: The E-Learning Use Case Diagram (Use case descriptions can be seen in* [*Appendix 1*](#_heading=h.oo36hsbsccxg)*)*

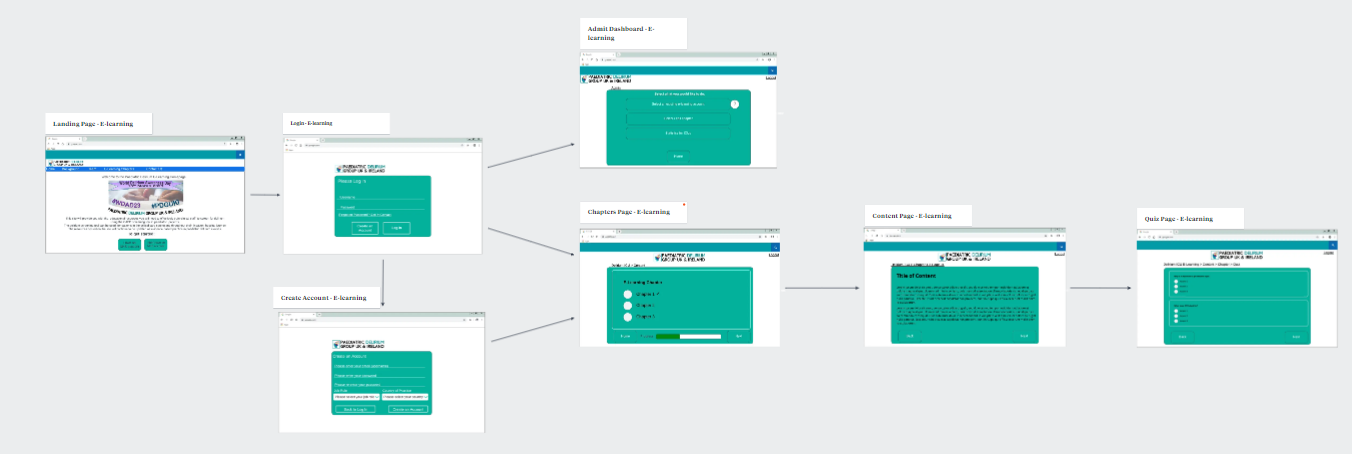
## 2.3. UI Design

The team had discussions during the start of the development of the prototype around what to do when it comes to the development of the pages of the site. This involved how everything would be connected, including the order in which the pages would be accessed and how the Audit and E-learning pages would be separated. There were also discussions regarding the design and how development would be carried out. To solve the issue of the design and the development of the pages, members of the team spent time to create wireframes of the pages so when it came to putting everything together, the whole team would have an understanding of the design, and everything would be consistent [(A2.2](#bookmark=id.a8cf9hwa5tja) - [A2.5)](#bookmark=id.dyl0b33yij3c). It was also decided that it was best to mainly use React components when making the pages so each one is standardised but still has the level of customisability that is required for them to be placed on multiple different pages. This solution saves the team time and prevents them from rewriting the same objects repeatedly.

The team was also discussing the design of the site when it came to the layout and colour scheme etc. of both the Audit and E-learning systems as this would be a great way to help the users move over to this new system because it would be a familiar setup and design. With the help of the Project champion and some discussions, it was decided that the colour scheme would stay the same as the current system (REQ-GUI-2). The team also decided to keep the same layout between the systems to keep a level of consistency that would avoid any confusion among the users. This also follows the [requirements](#_heading=h.etc0xq1fdkn0) to use the same colour and logo and to keep the design between systems the same (REQ-GUI-2).

To ensure that the system can be used by anyone, regardless of any sort of disability, the team wanted to ensure that there are adequate accessibility features that could be used. This would ensure that access to the system is as fair as possible, as anyone may struggle using the system and accessibility features will allow people with any issues that may prevent them from using the system wherever and whenever they like. This is to fall in line with the [accessibility requirements](#_heading=h.etc0xq1fdkn0) for the system (REQ-ACCESSIBILITY-1).

*Figure 2.3.1: The Audit Storyboard*

*Figure 2.3.2: The E-Learning Storyboard*

# 3. Implementation

The team's prototype implementation currently has a few key features with functionality which will be discussed in detail below. The group focus was to try to completely implement a small amount of the application's features as opposed to a large number of features with very limited functionality.

## 3.1. Features Implemented

### *3.1.1. Audit Log In*

#### Feature Description

The current prototype implementation features a fully functional Login system which includes both an initial front end design and the backend capabilities. The Login system works through API calls to check and retrieve data from the database for user authentication, as well as attaching roles to a user's session for the Role-Based-Access-Control (RBAC) [[4]](#bookmark=kix.8wvh3fnf6y7i) to different areas of the web application.

#### Reason for Prioritisation

The reason for prioritising this feature was due to its integral role as part of the team's web application. There are also multiple other features in the proposed web application that have dependencies on user roles and user authentication. After discussing these matters as a team concluded that rather than faking these items for future development it was an advantageous option to prioritise this feature for development.

### 3.1.2. Postgres Database

#### Feature Description

For the prototype the team decided to implement both the audit and e-learning Postgres databases, with some ‘dummy data’ for testing purposes. The team implemented the database structure that they intend to use for further development; Create, Read, Update and Delete (CRUD) APIs required to retrieve and insert data into the database and the roles necessary to access the database. The team decided to not implement the backup system for the database for the prototype as the team believed that to be out of reasonable scope within the designated time frame.

#### Reason for Prioritisation

The reason for the Postgres Database taking a priority position in the development backlog was its integral role for other features in the application’s functionality. By having this database implemented it will save time in future development by not having to create dummy data to fulfil feature dependencies that would later have to be refactored. This database can also be considered a stand-alone component of the system meaning future changes of other components will not affect the database.

### 3.1.3. Audit Graph Visualisation

#### Feature Description

The team made the decision to partially implement the Audit graph visualisations for the easy ingestion of the PICUs audit data. The implementation is of a fully functioning line graph (REQ-CONTENT-2) built from data that is retrieved through API connections to the Audit database. The reason this isn’t a complete implementation of this feature is due to not having the other graph and chart designs the team plan to develop in the coming sprints.

#### Reason for Prioritisation

The team's reasoning for prioritising this feature is due to its high priority in the project's requirements (REQ-CONTENT-2) as well as having a working prototype implementation of this feature helped the team to demonstrate a full user path of the Audit system for a PICU nurse submitting audit data. This partial implementation allowed the team to establish a solid base to more easily implement the other types of visualisations they plan to deliver as the implementation process should be similar.

### 3.1.4. Audit Data Input Form

#### Feature Description

For the prototype implementation the team has developed a complete data input form. This form takes ‘Yes’ or ‘No’ values for the information required to complete the Cornell Assessment of Pediatric Delirium (**CAPD**) [[5]](#bookmark=kix.sn1ovb87pf5y) and Sophia Observation withdrawal-Symptoms Paediatric Delirium (**SOSPD**) [[6]](#bookmark=kix.498e7eza8znk). The form is mostly implemented including its transfer of information collected to the Audit database through an API; The only measures left to be implemented are that of minor aesthetic means and miniscule functionality.

#### Reason for Prioritisation

Implementing this feature early in development allowed for the team to establish the APIs needed to input audit data into the audit database, which then helped with the development of other features such as the audit data visualisations as they could be built using verified data. This feature is also a key step in a typical user path of a PICU conducting their regular tasks.

## 3.2. Technology Components

Due to the complexity of the implementation, there is a substantial number of technologies used; it would be impractical to discuss every single one of them in depth. Therefore, the team has handpicked some of the most vital technologies used in the implementation for discussion in the table below.

*Figure 3.2.1: Current Technologies in Prototype*

|  |  |  |
| --- | --- | --- |
| **Technology** | **Utilisations** | **Discussion** |
| React / Bootstrap | Web application page components | React [[7]](#bookmark=kix.r3sdgiyumf19) uses a virtual Document Object Model (DOM), this is a lightweight representation of the actual DOM, this means a high level of performance is achieved. This approach gives React the capability to efficiently update and render its components without affecting the entire page.  As with most website solutions the web application will need a structuring markup language to use alongside React.  React also has a strong active community of other developers that the team will be able to utilise for support during development of the implementation through the multitude of forums and other resources available.  Unfortunately, previous implementations have been developed using Angular, this means the team used a different technology for the implementation. However, React's higher performance due to the virtual DOM and its considerations for mobile development that help with the implementation’s requirements for mobile devices (REQ-GUI-1) mean this is a worthwhile drawback. |
| Typescript | Web application functionality,  Used alongside React to dynamically customise components | The development team for the web application have decided to implement TypeScript as the main functionality language as opposed to JavaScript. The main reasoning for this is that the development team’s main experience is with Object-Oriented Programming languages, which through TypeScript’s strict type checking, more closely resembles the statically typed languages the development team have experience with. This reduces the learning curve the team has to overcome and therefore increases the development speed and effectiveness of the implementation. Furthermore, TypeScript's type checking functionality allows for the team's developers to detect and catch many errors at compile time, thereby enhancing the reliability of the code.  Another benefit of using TypeScript in the application is its ability to integrate seamlessly with React’s component based architecture and enable the development team to create dynamic custom React components whilst following best practice through the static type checking capability of TypeScript. TypeScript code can be easily integrated into an existing JavaScript codebase. This easy integration will allow the development team to build alongside existing JavaScript code used in previous implementations and utilise external sources with minimal refactoring. |
| **Technology** | **Utilisations** | **Discussion** |
| JEST.js | Testing | Jest is a JavaScript testing framework that is primarily used to test JavaScript code (can be used for TypeScript), in web applications such as the team's implementation.  Jest is optimised for speed and can run multiple tests in parallel, this allows for rapid testing of the web application, perfect for the rapid development style of the prototype implementation. Jest also provides a system that allows the team's testers to mock dependencies required for the application’s TypeScript code and therefore isolate its tests from external factors. As optimisation for speed is not a characteristic of the Jest framework this means for extremely large applications Jest may not be able to handle the large test suites and code with many complex dependencies and other options may be more suitable. For the size of this implementation this factor should not be of concern for the team.  Jest focuses on unit testing and is great for testing individual units of code, this limits the coverage of testing for the overall functionality of a web application. For the most robust solution the team is required to complete further methods testing such as Black Box testing referencing the requirements of the web application. |
| Node.js | APIs | For the APIs in the implementation the team have used the Node.js library which is a popular JavaScript runtime environment that allows the implementation to execute JavaScript code on the server-side, particularly APIs. For the implementation the team decided to also utilise the express.js framework which is specifically designed for creating and calling RESTful APIs during backend web application development.  For security measures when using APIs the team implemented password-hashing before the login system’s API calls for user verification. JSON web tokens (JWT) were applied alongside this to enable the team to enforce endpoint authentication for the different user roles in the system. |
| Docker | Used to ‘containerise’ the application | Docker gives the team the ability to create, deploy, and run the web application in environments called containers. Containers are lightweight, portable, and isolated environments that can be easily moved between the development, testing (local machines), and production (a QUB server) environments. As a docker container provides a consistent environment for the application to run, there is a level of confidence that the application will work as intended either on a Linux or Windows machine.  Docker has a simple and intuitive command-line interface that allows the development team to easily create, start, stop, and manage containers, there is also a GUI through the desktop application that provides similar functionalities. However, Docker is a relatively complex technology for those without experience to understand. Prior to this project the team had zero experience with Docker and its capabilities and therefore requires time and effort to fully master.  As Docker containers are isolated from each other, security is improved by not allowing one container to affect another. Docker also provides us the option to easily scale the application [[8]](#bookmark=kix.1dp0s0kp2la4) by creating multiple instances of a single container, it is likely this won’t be utilised during this development cycle however for future sprints this is an important factor. Docker is another external technology used for the implementation, in the future it may require additional configuration [(A2.1)](#bookmark=id.ap8h6ppebkqv) and maintenance, which can increase the repeating workload for the team responsible. |
| PostgreSQL | Database solution | The team decided on using PostgreSQL as it allowed them the option to use either a relational or non-relational database while using the same software and it also supports MVCC (Multi-Version Concurrency Control) [[9]](#bookmark=kix.rgjwbcg9x4la). This feature allows for the database to be configured so it can be used by multiple users at the same time and prevent errors during use. This fits with the project requirements to prevent any errors or bad data when two users try to submit forms at the same time, or someone views the data before submission has been completed (REQ-SCALABILITY-5). |

## 3.3. Technology Choice Reflection

When reflecting on the first technology choices made from the initial design of the system, most of the technologies previously stated are used in the current prototype implementation.

*Figure 3.3.1: Previous Technology Choice Reflection*

|  |  |  |
| --- | --- | --- |
| **Previous Choice** | **Alternative Used** | **Reasoning** |
| D3.js | react-chartjs-2 | In the previous implementation report the team decided that they would use the D3.js library for the audit data visualisations. Upon further investigation into what is required for the visualisations (REQ-CONTENT-2), it was discovered that D3.js was an over complicated technology choice and the decision was made to switch to the simpler chart.js library. The team are required to only give simple graphs and charts to represent this data, to which the team found that react-chartjs-2 was sufficient technology which would be simpler to utilise. |

## 3.4. Forecasted Technology Choices

It is impossible to predict the exact future, however, the paragraphs below discuss the technologies the team is deliberating on integrating into the final web application. This discussion will cover what the team believes they can use these technologies for in the final implementation as well as the benefits they may facilitate.

### 3.4.1. PuTTy for server

In order for the NHS and other affiliated parties to be able to access and utilise the web application the team need to place the application code files onto a private Queen’s University Belfast related server which can then be accessed through the public internet. The discussion on this subject has concluded with the decision to use the PuTTy software to establish a connection with the server. The advantage of this is that it allows the team to connect with the server through a Graphical User Interface (GUI) before interacting with it through a Linux terminal. Furthermore, the team will use the WinSCP protocol to transfer the application files to the server through the connection established in the PuTTy environment. This protocol greatly simplifies the file transfer method, ideal for this project’s implementation.

### 3.4.2. HTTPS

An advantageous future technology choice discussed among the team was to switch from using the HTTP protocol for the endpoints of the application to using the HTTPS protocol. This would modernise the solution as well as future-proof it for later development. The main advantage of this change for the application is that HTTPS is the secure version of HTTP and uses encryption methods to hide packet contents in network traffic from the application to external hosts. This would increase the security of the application and the application’s users as it would encrypt the traffic from any malicious third parties that may be looking into the packet traffic with ill intent [[10]](#bookmark=id.a9eewxz4x9fz).

## 3.5. Third-Party Components & Code

### 3.5.1. Use of Bootstrap

For development of the web system the team decided to make use of the unlicensed open-source Bootstrap framework. Bootstrap gives the team access to an extensive range of pre-designed HTML, CSS, and JavaScript (can be adapted for TypeScript) components that can be easily customised and used to suit the system’s needs. By using Bootstrap, the team is able to rapidly create a professional web application with consistent components easily. The use of Bootstrap in the prototype implementation includes components such as navigation bars, buttons and large modals.

### 3.5.2. Use of External Resources

Alongside development of the prototype application the development team used external resources to share knowledge amongst the team members and help decipher issues they faced. These external resources include public help forums from websites such as Stack Overflow as well as tutorials from the basic to advanced examples of the technologies used in the application. The team used these resources as reference points to aid their personal development with unfamiliar technologies and to help solve common development speed bumps that other software engineering communities have faced.

# 4. Testing

## 4.1. Code Test Plan

### 4.1.1. Introduction

The following test plan will be used to identify any bugs in the team’s prototype to ensure that it is robust and fit for use. This test plan outlines our approach to testing and improving the prototype before the next stage of development. The main goal is to test as many of our functional and non-functional requirements as possible.

### 4.1.2. Testing Approach

The team has decided to use white box testing, specifically unit testing, to verify that the system's main functionality operates correctly. The team believes that conducting unit testing will lay the foundation for ensuring the system performs as intended. This testing methodology is designed to take the path of least resistance to validate the fundamental software requirements.

In addition to unit testing, they will also employ coverage-based testing to ensure that all code paths and branches are thoroughly tested. The team’s objective is to achieve a 90% branch coverage and a 95% line coverage. This approach aims to verify that every line of code functions as intended, leaving no stones unturned, and any missed items in the unit testing are found. To perform these tests, the team has opted to use Jest, a JavaScript testing framework.

### 4.1.3. Scope

Below is a basic overview of the categories we will be testing.

*Figure 4.1.3.1: Testing Scope*

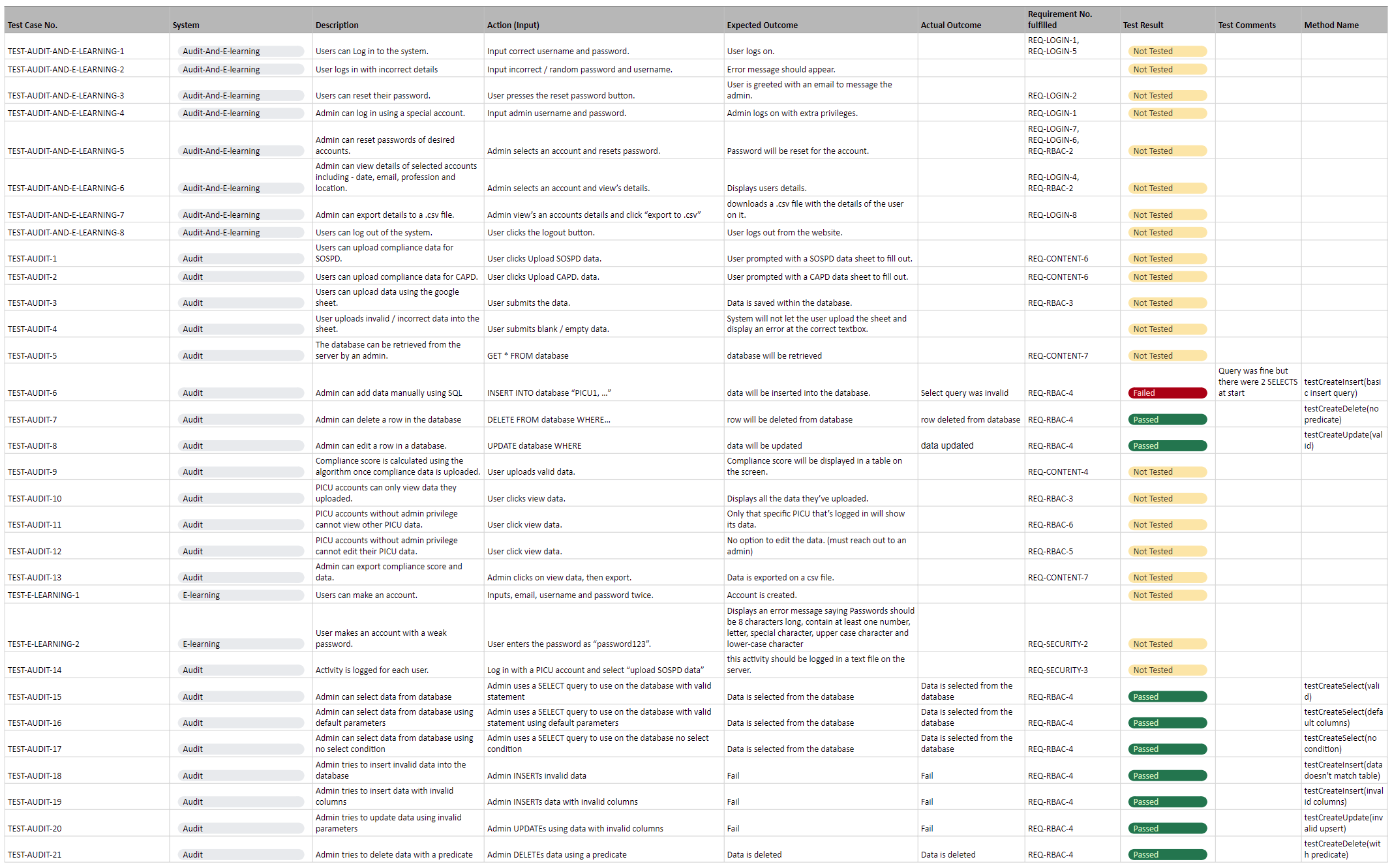
|  |  |
| --- | --- |
| **Category** | **Description** |
| GUI | Consistency throughout the theme and of the webpage and ensuring popular devices (smartphone, laptop) can access the site. |
| Content | The functionality of the site must enhance the content provided by the stakeholders |
| Login | Logging into the site is a key feature in ensuring a connection can be made with a secure user. |
| RBAC | RBAC ensures that admins can access key parts of the site that other users cannot. |
| Scalability | Future proofing to make sure that if the system grows the website will be able to handle it. |
| Security | Activity throughout the site will be logged and a history of changes to the data should be saved. |
| Visualisation | Graphics of audit data must be correct and display data in an easily digested way. |
| Accessibility | Make sure that the site can be used by users with accessibility issues, for example, high contrast mode. |
| Maintainability | Staff without technical experience should be able to edit and make changes to data if they have the correct privileges. |
| Backup | The database must be maintained and backed up regularly. |

## 4.2. Code Testing

### 4.2.1 Unit Testing

Below is a unit test plan based on the requirements to ensure that the specification is met for our prototype. Each test case will cater towards a requirement. “Unit testing is the process of testing program components, such as methods or object classes. Individual functions or methods are the simplest type of component.” [[1]](#bookmark=kix.mzg9l6kyh8dj)

*Figure 4.2.1.1: A Selection of Test Cases*



### 4.2.2 Coverage Based Testing

The team’s test plan for white box testing aimed to achieve the highest possible line and branch coverage possible. This was achieved by examining the code with the goal of improving them and by adding more test methods. “Code coverage in principle, every code segment that you write should have at least one associated test. Therefore, you can be confident that all the code in the system has actually been executed. Code is tested as it is written, so defects are discovered early in the development process.” [[1]](#bookmark=kix.mzg9l6kyh8dj)

White Box testing was used to make sure that all independent paths within a module have been exercised at least once. This means all statements need to be covered to see if they are true or false and all loops are executed in full. Below is an example of testing code coverage of our prototype, this is just for sake of demonstration as time constraints meant the team couldn’t test the software in full as it is just a prototype.A picture containing text, screenshot, colorfulness

Description automatically generated

*Figure 4.2.2.1: Prototype Test Coverage*

### 4.2.3 Code Reviews

“Code review is a process in which code is reviewed by someone other than the author, often before the introduction of that code into a codebase.” [[11]](#bookmark=kix.c12w9zr4f3y3)

Throughout our implementation of our prototype, before the team merged their personal branches they always reviewed the code as a group. During [peer programming sessions](#bookmark=id.2ld78asjtvmu) [(A2.6](#bookmark=id.2ld78asjtvmu) - [A2.7)](#bookmark=id.pos6nhdl1d0g) the team would review the code produced at the end of the session. This way the team could maintain consistency throughout development and know which branches were used for a particular feature.

“Some of the benefits of code review, such as detecting bugs in code before they enter a codebase, are well established.” [[11]](#bookmark=kix.c12w9zr4f3y3) Reviewing code helped with consistency throughout the code and made sure the entire team was engaged in development.

## 4.3. User Evaluation

As an extended precaution the team had peers within the module (outside of the team) come in to help test the prototype to ensure it meets the expectations. User evaluation helps identify defects and functional gaps, ensuring that the final product satisfies user needs and requirements. “Production based testing makes it possible to collect a lot of data about user behaviour.” [[11]](#bookmark=kix.c12w9zr4f3y3) This would simulate end-users testing the prototype as if it were released.

A member of the team acted as a guide, showing the subjects how the system worked by sharing their screen. Observers could ask questions and steer the guide into showing them parts of the system. The team created a google form for the observers to describe their thoughts / opinions on the prototype.

The form responses can be viewed in detail [here](https://docs.google.com/spreadsheets/d/1qICKQKSHpgxdr0MZ3QJ2j-z1bwESV9CmYG3CEADKj9k/edit#gid=1784722418).

**Comments from Dean Logan (PwC Colleague and Peer):**

As Dean is a fellow peer of the team, he gave a lot of useful information offering his insights on how the site could be improved from a technical and user friendly perspective.

Dean’s comments on the GUI:

* “Logo looks squished and the text is too small.”
* “The navigation bar doesn't appear to be centred, it looks like there is a slight lean to the left side of the screen.”
* “The chart that's displayed on the graph page is too small, lots of empty screen space that could be used here on every screen size.”
* “Font type and colour used are clear and legible.”
* “The dropdown on the chart page works well, the additional ability to search/filter the different types of charts is a great feature.”
* “The home page is not consistent at all with the rest of the website. The home page should be contained within a ‘tale’ coloured box with white text, similar to the rest of the pages on the website.”
* “The navigation bar should be fixed to the top of the screen at all times, this will avoid the user having to scroll to the top of the page.”

Dean’s comments on the logic:

* “Whenever a user is logged in they should not be able to log into another account again by pressing the login button (or navigate to this page through entering it into the URL), this should be replaced with a logout button whenever a user is logged in.”
* “There should be an alert to notify the user when they have logged into the system and somewhere on the website where the user can see what account is currently logged in (e.g. username of current user displayed in the corner of the screen or in the nav bar).”
* “Overall the system does look good with only some minor complaints mainly surrounding sizing for 24 inch screens (1920x1080) and above where there seems to be large amounts of whitespace, which for a prototype really is a minor complaint however in the finished system it would be great to see better utilisation of this space and it works well on mobile.”

**Comments from Lisa McIlmurray (Medical PhD Candidate and Project Client):**

Lisa was very pleased with the overall system especially the theme and colourway describing it as “well established project branding” also saying the site was “easy to use” and “very self explanatory” in nature. She mentioned that a “how to use” guide could be useful for less tech-savvy people.

**Comments from Charles Gillian (Senior Professor and Course Lecturer):**

Charles was also very happy with the system at hand saying the theme showed a “Nice choice of colours” and it was well presented too. Charles described the system as a “Good Solution at hand” which is very encouraging coming from him as an expert within the field.

In conclusion the team is very happy with the general feedback we’ve received. People are content with the prototype and feel like it’s fit for purpose. This is very encouraging as the software is in early stages of development. Dean’s comments were helpful as they critiqued the system from a software developer’s perspective and will definitely be reviewed when developing the entire system next year.

# 5. Project Management: Roadmap and Sprint Plan Updates

## 5.1. Project Management Tools

### 5.1.1. Managing Documentation and Meetings

It was decided, at the beginning of the project, that the team would primarily use G Suite [(A2.8)](#bookmark=id.lfybqerd96u9) as the primary tool for collaboration. One key reason for this is due to the groups greater experience using these tools, due to our time on placement with PwC, over similar tools such as Microsoft Teams.

A shared Google Calendar was created to schedule team meetings and to remind each member of the team of both the internal and external deadlines. The shared Google Drive is used to share ‘non-code’ related documents, such as those related to the report. Due to its integration with Google Drive and compatibility with ‘.docx’ files, Google Docs was chosen to edit and create documentation. Additional benefits of this platform include the ability for multiple users to work concurrently on a document, ‘suggestion’ mode allowing the changes to be made without committing them, being able to leave comments and assigning tasks to individuals with email notifications. A single Google Sheet was used to manage the requirements, use case descriptions and the test plan allowing for more meaningful IDs to be created and for when these IDs may change for them to be referenced more easily. Other advantages of using a spreadsheet to record this data include greater analysis of this data and ensuring data consistency. Another advantage of G Suite applications is the built in version control, in which changes, and who made them, can be tracked and if necessary, these older versions can be reverted too.

A disadvantage of using this platform is that we must ensure the compatibility of our reports within the ‘.docx’ file format, as this is used by the examiner. Another disadvantage is that the project champions use Microsoft Teams for video calls and therefore the same integration with the documentation and the meeting cannot be used.

To collaborate when working on the diagrams used for design, LucidChart and diagrams.net were used to allow for the same concurrent use provided by Google Docs.

### 5.1.2. Managing Development

To manage the development of the system a mixture of Jira and Google Sheets were used. Google Sheets is used in the small, yet still important, aspect of keeping track of the state of each individual requirement, whether it has not been started, still in progress or has been completed.

The use of Jira was primarily to delegate, manage and review individual’s workloads through what is known as ‘issues’ which translate to both singular tasks and a collection of tasks regarding development and documentation. These tasks are grouped together by ‘epics’ which represent a larger task or deliverable, in which the deadlines for each ‘epic’ is mapped onto the road map which can be seen in [figure 5.2.1](#bookmark=id.gtcvvgun3dan). Each issue is assigned to a team member to allow the whole team to be aware of the work being completed by others, and the state of said work.

One disadvantage of using Jira is that it adds overhead to the work, which is being completed, as time is required to be invested in ensuring that the state of the tasks is accurately represented. Time must also be spent to decide which tasks relate to which group, which again distracts from developing the system.

The chosen source control technology was Git, along with GitLab for the repository. As Git is the industry standard this was perhaps an easy choice for the team to have made. Due to the ability to track each team member's progress, revert to previous versions and the ability to merge, this was extremely beneficial within development.

### 5.1.3. Links for Online Work Environment

* [Jira](https://svspaediatricdelirium.atlassian.net/jira/software/c/projects/SVSP/boards/2)
* [GitLab](https://gitlab2.eeecs.qub.ac.uk/40293585/svs-paediatric-delerium)
* [Project Sheet](https://docs.google.com/spreadsheets/d/1qICKQKSHpgxdr0MZ3QJ2j-z1bwESV9CmYG3CEADKj9k/edit?usp=sharing)
* [Shared Drive](https://drive.google.com/drive/folders/1MutBd-loI6WYAPUdZSvFqrsYtAgN18nm?usp=sharing)

## 5.2. Gantt Chart

The team designed a Gantt Chart to help organise our future development plan for our prototype. This Gantt Chart will be for SDP 2 and covers from September – December. It goes into detail of what we need to complete in the future.

*Figure 5.2.1: Gantt Chart*

A picture containing screenshot, text, line, colorfulness

Description automatically generated

## 5.3. Sprint Plan

### 5.3.1. Sprint Description

For the Sprint 4 period, the team plans to enhance existing features from the prototype implementation they have developed over the course of CSC3068.

One of the first enhancements the team will implement to the system is improvements to the Audit Login system. The team will include features such as the ability for a user to request a password change as well as added security measures on the acceptance of these passwords. New passwords will have to include at least 8 characters, 1 numeric character, 1 upper-case character, 1 lower-case character and 1 special character.

A further enhancement the team will implement to the prototype system will be to the graph visualisations. The enhancements brought to this feature will be extended ways to view the audit compliance data through further chart designs, as well as the user having the ability to compare their own PICU compliance data against the rest of the compliance data held on the system.

A new feature the team will implement to the system is the Admin / Field Engineer role’s functionality. These users require the ability to be able to edit and export the audit compliance data, change a user account’s password and create new user accounts.

For security and system admin reasons, the team will implement an audit logging system that will only be accessible by users with an admin role. This audit log will document all the changes to the Audit database including data being added or removed as well as the user committing the change.

### 

### 

### 5.3.2. Reviewed Risk of Features

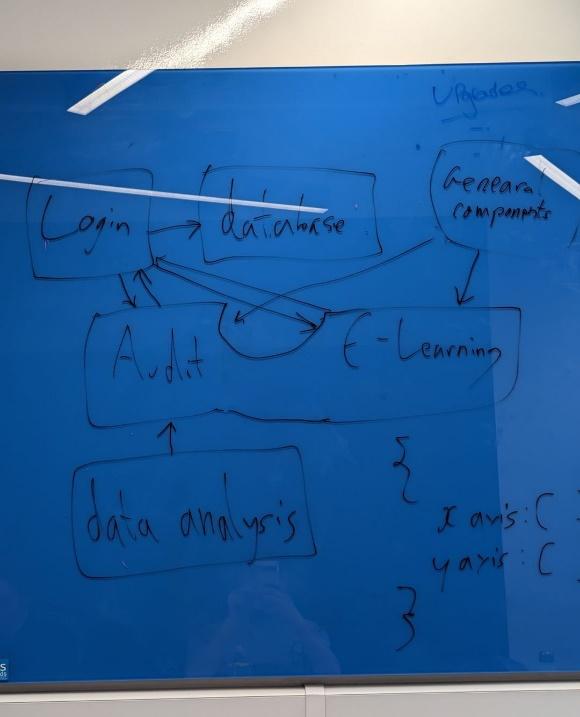
*Figure 5.3.2: Risk of Features Breakdown*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Description** | **Risks** | **Mitigation** | **Acceptance Measure** |
| Improved Login System | Improved security on user accounts is a priority. This would mean added password restrictions such as required characters and length and the addition of the ‘change password’ feature if anyone forgets or loses their password. | The added complexity of the password requirements may mean that users are more likely to forget their password and require it be changed or reset. | The addition of the change password feature resolves the risk of people forgetting passwords because it allows the user to get a new password, removing the risk of them not being able to access their account. | Having a Login system which requires passwords to be at least 8 characters long with at least 1 numeric, upper-case, lower-case and special character each along with a feature to change an account’s password |
| Increased Audit Visualisations | Further development will be made to the existing visualisation, such as including different types of graphs and better support for mobile devices.  A new visualisation, in regards to overall compliance of PICUs, will also be created with the same functionality as the above visualisation. | When comparing PICU data sensitive data needs to be anonymised. Comparisons between PICUs should follow GDPR guidelines. | An action that is already taken, which mitigates the risk, is by shuffling the overall, using the Fisher–Yates algorithm, compliance data into a random order and therefore a single user is not able to identify which compliance score belongs to a specific PICU patient.  By representing a patient as a bed number within the system this allows for the identity of the patient to remain hidden. | Having a visualisation page that allows a user to select multiple chart types. Upon selection of the chart types the visualisation will change to correlate with the specified type. |
| Admin / Field Engineer Functionality | Admins and field engineers need their functionality implemented. They will be able to edit and export compliance data, along with being able to reset passwords for accounts and create new accounts. | The admin and field engineer page needs to be secured as sensitive data can be downloaded and serious changes can be made | Make sure the admin and field engineer pages are secured and only an admin and field engineer can log into it. | An admin and field engineer page with the ability to change passwords, edit and export compliance data and create new accounts. |
| Audit Logging | The team has decided to log every action made on the website. Actions will be logged to a downloadable file. This improves security as the attacker's actions will be recorded and therefore allow identification of the attacker. | The data being logged is very sensitive and the integrity of this data may be compromised by attackers attempting to mask their activity.  The logs may also take up a lot of storage space on the servers if they are backed up regularly. | Ensuring that the log file is secure, possibly through encryption.  A rolling deletion method which deletes logs which are of a certain age, such as 30 days. | Major activities which are conducted by users are recorded along with the time, username and user role |

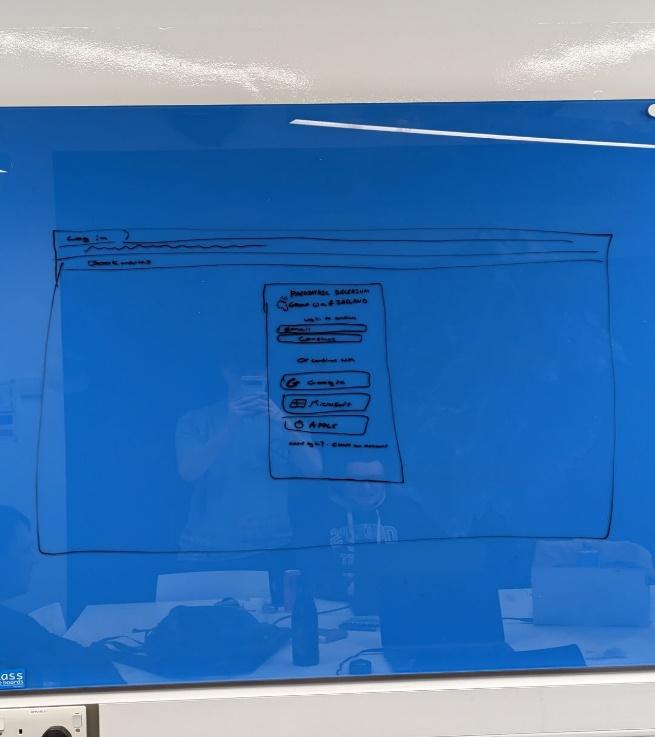
# 6. Appendices

## Appendix 1: Use Case Descriptions

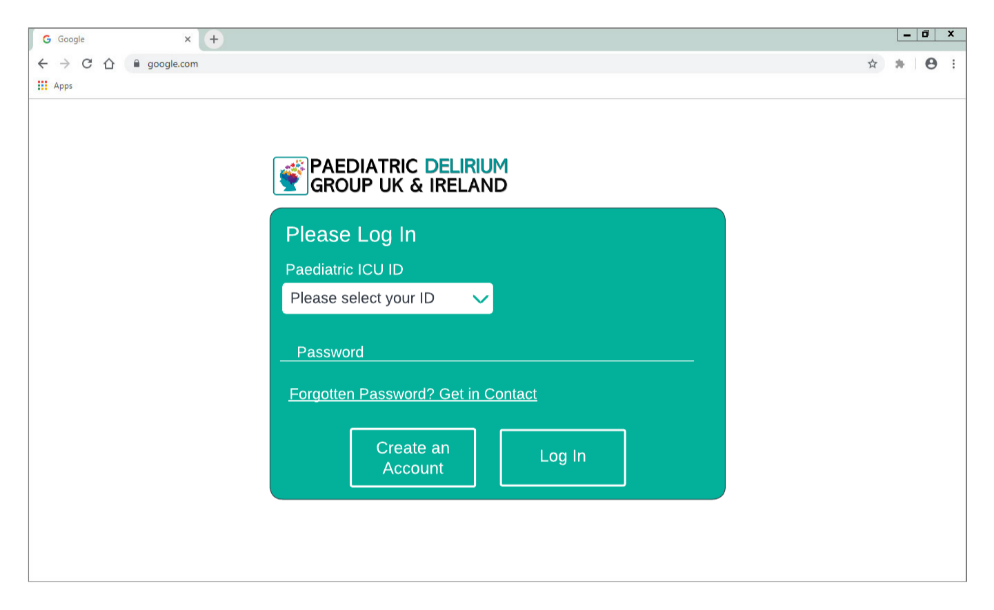
## Appendix 2: Photos



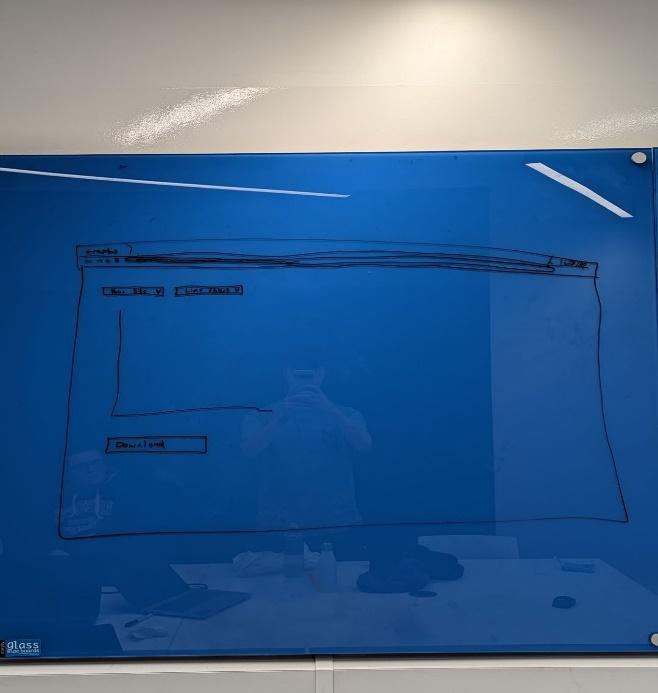
*Figure A2.1: Hand-drawn Docker Architecture Sketch*



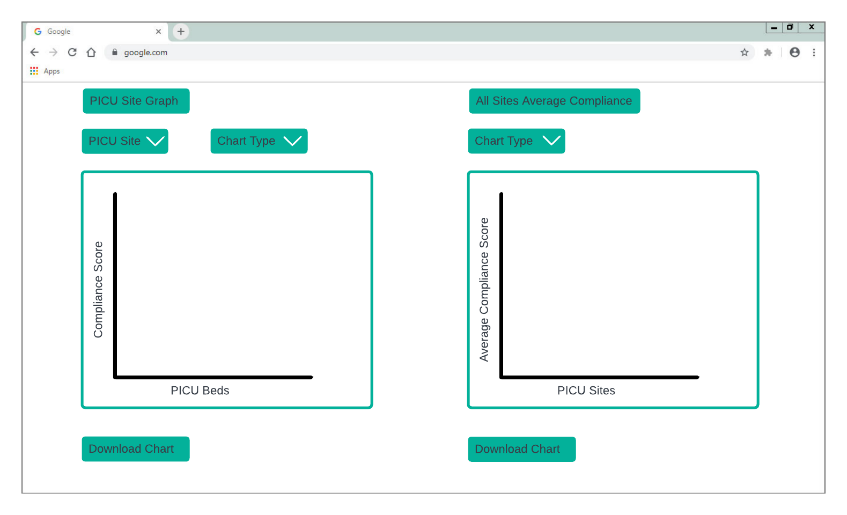
*Figure A2.2: Hand-drawn login page wireframe*



*Figure A2.3: Digital login page wireframe*



*Figure A2.4: Hand-drawn Audit visualisation wireframe*



*Figure A2.5: Digital Audit visualisation page wireframe*



*Figure A2.6: Robbie Duncan (Left) and Ewan Forsythe (Right)*

*collaborating on the prototype*



*Figure A2.7: Adam Logan (Left) and Andrew Robb*

*(Right) collaborating on the prototype*

# 

*A2.8: Online collaborative session*

# 

# 7. References

1. I. Sommerville, *Software engineering 10th Edition*. Pearson Education, 2016.
2. D. T. Haley, B. Nuseibeh, H. C. Sharp and J. Taylor, "The conundrum of categorising requirements: managing requirements for learning on the move," Proceedings. 12th IEEE International Requirements Engineering Conference, 2004., Kyoto, Japan, 2004, pp. 309-314, doi: 10.1109/ICRE.2004.1335688.
3. The Open Group, Service Oriented Architecture (SOA) in the Real World, Microsoft Press, 2007
4. Ravi S. Sandhu, Role-based Access Control11Portions of this chapter have been published earlier in Sandhu et al. (1996), Sandhu (1996), Sandhu and Bhamidipati (1997), Sandhu et al. (1997) and Sandhu and Feinstein (1994)., Editor(s): Marvin V. Zelkowitz, Advances in Computers, Elsevier, Volume 46, 1998, Pages 237-286, ISSN 0065-2458, ISBN 9780120121465, <https://doi.org/10.1016/S0065-2458(08)60206-5>.
5. Traube C, Silver G, Kearney J, Patel A, Atkinson TM, Yoon MJ, Halpert S, Augenstein J, Sickles LE, Li C, Greenwald B. Cornell Assessment of Pediatric Delirium: a valid, rapid, observational tool for screening delirium in the PICU\*. Crit Care Med. 2014 Mar;42(3):656-63. doi: 10.1097/CCM.0b013e3182a66b76. PMID: 24145848; PMCID: PMC5527829.
6. Ista E, van Beusekom B, van Rosmalen J, Kneyber MCJ, Lemson J, Brouwers A, Dieleman GC, Dierckx B, de Hoog M, Tibboel D, van Dijk M. Validation of the SOS-PD scale for assessment of pediatric delirium: a multicenter study. Crit Care. 2018 Nov 20;22(1):309. doi: 10.1186/s13054-018-2238-z. PMID: 30458826; PMCID: PMC6247513.
7. BOOK, React.js Essentials, Fedosejev, A., 9781782174622, https://books.google.co.uk/books?id=Rhl1CgAAQBAJ, 2015, Packt Publishing
8. M. T. Chung, N. Quang-Hung, M. -T. Nguyen and N. Thoai, "Using Docker in high performance computing applications," 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha-Long, Vietnam, 2016, pp. 52-57, doi: 10.1109/CCE.2016.7562612.
9. Bernstein, P.A. and Goodman, N., 1983. Multiversion concurrency control—theory and algorithms. ACM Transactions on Database Systems (TODS), 8(4), pp.465-483.
10. F. Callegati, W. Cerroni and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," in IEEE Security & Privacy, vol. 7, no. 1, pp. 78-81, Jan.-Feb. 2009, doi: 10.1109/MSP.2009.12.
11. T. Winters, T. Manshreck, and H. Wright, *Software engineering at google: Lessons learned from programming over time*. O’Reilly Media, 2020.