# H48W 35 Computing: Software Development Graded Unit 2

**Project Stage: Evaluation**

**Evaluation Report**

**Student name: Daria Vekic**

**Student number: 586661**

**Date: May 2023**

Table of Contents

[H48W 35 Computing: Software Development Graded Unit 2 1](#_Toc135854640)

[Introduction 1](#_Toc135854641)

[The Solution 1](#_Toc135854642)

[Functional Requirement 1 1](#_Toc135854643)

[Functional Requirement 2 5](#_Toc135854644)

[Functional Requirement 3 5](#_Toc135854645)

[Functional Requirement 4 6](#_Toc135854646)

[Functional Requirement 5 6](#_Toc135854647)

[Summary 7](#_Toc135854648)

[Strengths and Weaknesses 8](#_Toc135854649)

[Technical Documentation 8](#_Toc135854650)

[Inception Planning 8](#_Toc135854651)

[Solution Planning 9](#_Toc135854652)

[Development 11](#_Toc135854653)

[User Documentation 12](#_Toc135854654)

[Prototype Application 13](#_Toc135854655)

[Future Development Recommendations 15](#_Toc135854656)

[Data Persistence 15](#_Toc135854657)

[Sort Functionality 15](#_Toc135854658)

[Document Management 15](#_Toc135854659)

[Invoicing 16](#_Toc135854660)

[Summary 16](#_Toc135854661)

[Modifications and Unforeseen Events 17](#_Toc135854662)

[The Static Model 17](#_Toc135854663)

[Wullie 18](#_Toc135854664)

[Stirling University 18](#_Toc135854665)

[Other Modifications 19](#_Toc135854666)

[Personal Reflection 21](#_Toc135854667)

[Programming 21](#_Toc135854668)

[Ethical Hacking Tool 21](#_Toc135854669)

[Time to Face the Ternary Operator 22](#_Toc135854670)

[Meta-Skills 22](#_Toc135854671)

[Self‑Management 22](#_Toc135854672)

[Innovation 23](#_Toc135854673)

[Bibliography 24](#_Toc135854674)

# Introduction

This document relates to the Graded Unit 2 project as part of the HND Computing: Software Development H48W 35. It aims to satisfy Stage 3, Evaluation of the overall project by providing a report which reflects upon a variety of aspects of the project.

# The Solution

Overall, the solution successfully met the majority of the functional requirements that were identified in the Inception Planning Report. The final functional requirements can be found at page 51 of the Inception Planning Report, but I have included them below for ease of reference:

FR 1. The system should validate employee log in details.

FR 1.1. A username should be generated for new users.

FR 1.2. A password must conform to specific criteria.

FR 2. The system should be able to create a new case.

FR 2.1. Valid case details must be accepted.

FR 2.2. Valid client details must be accepted.

FR 2.3. Valid solicitor details must be accepted.

FR 2.4. A unique case reference number must be assigned to the case.

FR 3. The system should accept a case reference number and display its details.

FR 3.1. A case reference number must be in a valid format.

FR 3.2. The system should remove a case when necessary.

FR 3.3. The system should record billable activities entered by user.

FR 4. The system should generate suitable reports.

FR 5. The system should display all cases when necessary.

## Functional Requirement 1

This requirement relates to the use case “Log In”. It unsurprisingly suggested to me that the client wanted their users to be able to log in to their new system. It was outlined in the Development Report that this iteration would focus on developing this use case along with the “Add New Case” use case. I wanted to attempt to develop the log in system to a secure standard, therefore I did not cover generating usernames for new users. I can therefore say that this sub-requirement wasn’t met.

However, the Development Report acknowledges this and provides a suggestion of how this would be implemented in future iterations. On page 41 of the Development Report, there is the suggestion that rather than using a hard‑coded username, there could be an additional attribute in the Employee class (Employees are the main users of this system) along the lines of private final USERNAME. There is then a suggestion to include a method generateUsername() which might contain implementation along the lines of return firstName.toLowerCase() + "." + lastName.toLowerCase(). This method could then be called within the constructor of the Employee class, so whenever a new Employee is added to the system, their new username is generated at the same time. The use case “Add Staff Member” was not a focus of this iteration of development, and this was made clear in the Development Report.

The requirement of validating employee log in details was implemented successfully. I spent quite a lot of time on trying to satisfy this requirement as the steps required to validate log‑in details weren’t wholly obvious to me. So, initially this was broken down into smaller steps where a Map was filled with the Employee’s usernames (hard‑coded for this iteration), and a pre‑defined password. The first step involves receiving the username value and checking that the username exists in the Map. I have included screenshots of the code for this in the Development Report at Figures 58, 59, and 60. This prevents the system from allowing a user access if their input is not an exact match to the Key in the Map.

The next part of fulfilling this requirement was ensuring that the password entered by the user is the correct password. So if the username exists, ask for the password. If the password is an exact match to the Value in the Map for that Key, then the user is allowed access. Otherwise, the password is not valid. Again, screenshots of this have been included in the Development Report at Figures 61, 62, and 63.

Although this essentially satisfies FR1, it isn’t realistic to store passwords in plain text. So, at this point some research was undertaken with regard to password hashing algorithms. At the same time, some thought was given to the ability for a user to be able to change their password. After some discussion with the client, I moved forward with implementing the storing of a hashed password value and the new requirement of changing a password. FR 1.2 implies that the user should be able to change their password, but this wasn’t quite clear in Inception. This was clarified in meeting number 5 where the resetting of a password was demonstrated to the client, and the client confirmed that they are happy with allowing their employees to have this ability (see meeting number 5 minutes in Development Report, agenda item 6). With this in mind, I implemented this by making use of the replace() method to update the Map with the user’s new password. Before the Map is updated, the validation algorithm is performed.

It is simple enough to make use of abstracted code like the replace() method. But, to ensure the data is in a state we want it to be in, I developed the validation algorithm based on the minimum criteria outlined at Non-Functional Requirement 2 in the Inception Planning Report. This was also tested thoroughly through unit testing whilst being developed, and also through black box testing performed later in Development. One reason this validation algorithm worked successfully is that it was clarified with the client that the only special characters to be included were “standard” special characters (see page 46 of Development Report). This saved me unnecessarily wasting time and effort. So, I made use of the Pattern and Matcher class to perform a check on each of the client’s desired password criteria:

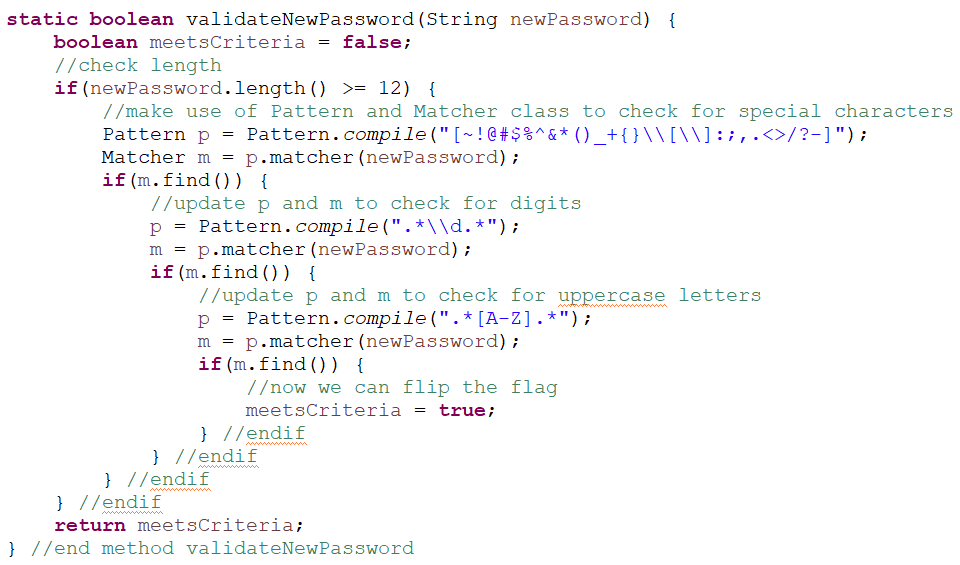


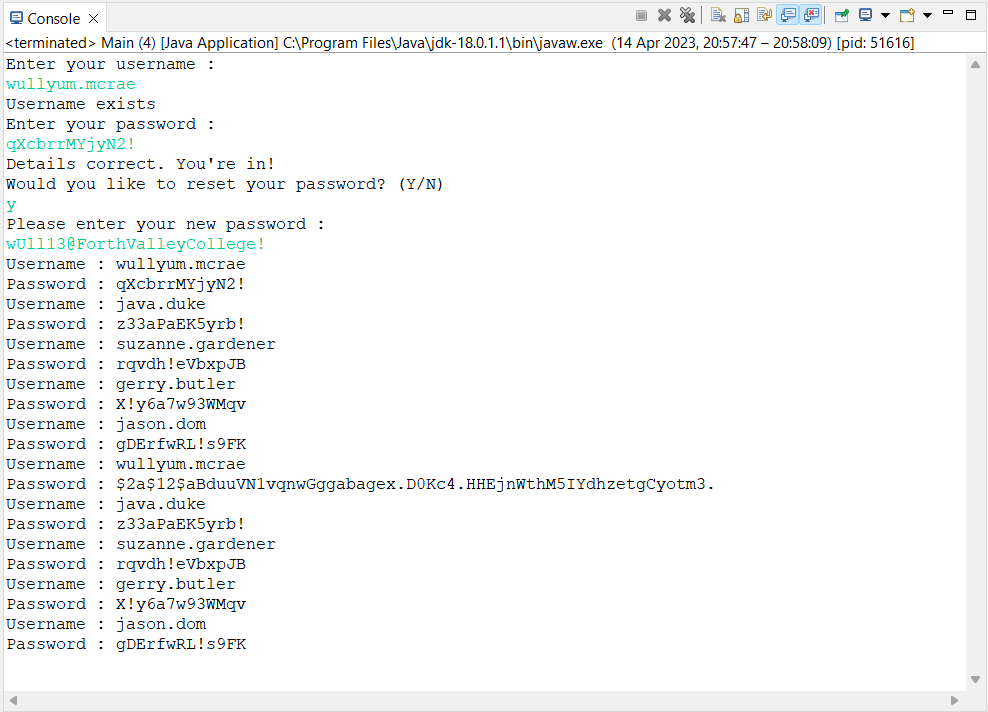
Figure Password validation

Checking for each criteria one step at a time contributes to the success of FR 1.2. With this, if the user’s new password fails to meet just one of the criteria, they are unable to reset their password. Now, the Map will only be updated once the user enters a new password that meets all of the criteria. The same logic of validating log in details applies thereafter as the user is returned to the log in window where they can log in using their username and new password: check the username exists, if it does, ask for the (new) password, if it matches, grant access.

To ensure that no password is stored as plain text, a portion of code was abstracted from somewhere on the internet. This required a fair bit of research and led to a section on password hashing in the Development Report that serves as unfamiliar territory for me. In the end, I opted to try and implement BCrypt in my Java program as it’s more suitable for hashing a password than alternatives like SHA‑512. It seemed easier to implement SHA‑512 as a password hashing function because it is listed under the algorithms that can be specified when using a MessageDigest instance (see Figure 71 in Development Report). However I didn’t want to blindly use an algorithm just because it is supported in Java’s MessageDigest class. This felt like a slight risk as I had no idea if trying to implement BCrypt in a Java program was going to be a waste of time and effort. However, after sourcing the code for the BCrypt algorithm and adding it into my own Java project, it appeared to work successfully. The BCrypt class was taken from:

https://github.com/lokeshgupta1981/Core-Java/commit/fcebd97daec176dc94effce88c077099c46900f0#diff-8e375029d3038f8953b53f71c37a30bd623e294fbbf5dcb2bea48f1dc7295366

And I then tested that it did indeed store only hash values:



Before wullyum.mcrae changed his password

After wullyum.mcrae changed his password – the **hash value** is now stored in the Map.

Wullie's plain text password

What's being stored

Figure Testing BCrypt

The security of BCrypt was explored a little more through putting a hash value through the password cracking tool hashcat in Kali Linux on a virtual machine. Everything pointed to opting for BCrypt over the algorithms listed under Java’s MessageDigest class (see Figure 71 in Development Report). This further exploration into password hashing algorithms paid off and contributes to the success of meeting FR1.

Furthermore, a variety of tests were performed on this requirement in addition to the unit testing that took place earlier in Development. The document “Test Plan.docx” submitted in Development indicates that test numbers 1 through 12 (inclusive) are included for the purpose of testing data validation and interface controls that relate to logging in. It is clear from the document “Test Log.docx” that these tests are all successful and pass the given test. They cover cases from logging in using normal data and testing for: a username does not exist, a password doesn’t match, the “Reset Password” link opens a new window, the Map is updated and allows user access with updated password, that only a hash value is stored, an updated hash value is stored after user resets password, along with testing each minimum password criteria in turn. Overall, discounting FR 1.1, FR 1 has been met successfully.

## Functional Requirement 2

This functional requirement relates to the use case “Create New Case”. This was the other requirement that this iteration of Development focused on implementing. I decided to implement this requirement as it has the most amount of user input in comparison to other requirements. I therefore wanted to concentrate time and effort on implementing this requirement well, rather than implementing all use cases to a lower standard.

In order to actually construct a new Case object, quite a lot of data is required to be validated and formatted first. This requirement was therefore quite substantial in the success of the system. Starting off Development by devising various validation routines contributed to the success of this requirement. It allowed me to clarify a few data rules with the client, such as the maximum length of a name, and also allowed for extensive unit testing early in Development. This meant that bugs and issues were discovered earlier and therefore enough time was available to rectify these errors and to fulfil the requirement successfully. Also crucial to the success of this requirement was the implementation of the Model, View, Controller organisational technique. Starting off by coding the validation methods lead to the question of where to include these methods in the program, which lead to researching another unfamiliar territory and resulted in a better organised program overall.

The generation of a unique case reference number was also implemented successfully. The constructor of the Case class includes a field private static int caseCount along with a field private final String CASE\_REF\_NUM. I made caseCount static so that no two Cases are allocated the same number, and CASE\_REF\_NUM is final because once a reference number is allocated to one Case, it should not be allocated again nor should it change. Since the case reference number will not change for any given Case object, we therefore want it to be immutable, therefore made final.

Although the system meets FR2 successfully, the usability testing revealed that an error message could have been made more informative. The system definitely checks for the length of a phone number, but the error message doesn’t indicate that the phone number field requires data that must be 11 digits long. This doesn’t mean the requirement has not been met; it has, but the user experience could have been made slightly smoother by including a more informative error message.

## Functional Requirement 3

This requirement has been fulfilled in part.

In terms of its corresponding use case, “Search Cases”, I felt the detail required to be implemented would have sacrificed quality in the other requirements. For example, a lot of time was concentrated on developing a secure log in system along with adding a new **and valid** Case. So, to avoid implementing every use case to a poorer standard, this functional requirement was not a priority in this iteration. Furthermore, in order for this requirement to be met in full, the requirement of adding a new Case absolutely had to be completed – there would be no Cases to search if they couldn’t be created. Similarly, you wouldn’t be able to remove a Case if the List of Cases is empty. Therefore, I implemented this requirement only partially so a user can search for a Case reference number and view its details. The relevant interface controls that would lead to deleting a Case and recording billable activities were included, but if these are pressed a simple message is displayed to the user indicating they will have this ability in future.

With this in mind, the part of this requirement that was implemented was done so successfully. There is testing included in the Test Plan for this requirement, to verify that entering a valid case number retrieves a specific Case and that invalid input displays an error message. The Test Log shows that these tests pass, and relevant screenshots are included at Appendix 30 and 31 in the Test Log.

There is not much else to say about this requirement. If the user searches for a Case, the system retrieves that Case. If the reference number entered isn’t found, an error message is displayed.

## Functional Requirement 4

This requirement relates to the generation of reports in the new system. However, this was not a priority for the client, and as such this requirement was not implemented at all in this iteration. This was agreed at client meeting number 3, the minutes of which can be found in the Solution Planning Report.

## Functional Requirement 5

This requirement was partially implemented. FR5 involves simply outputting the contents of the List of Cases to screen. Initially, I programmed this requirement to display the contents of the List in the console of the IDE. However, when it came to creating a jar file to deploy the system to client, this was no use as when the jar file is run, it doesn’t have the console to print the data to. It was time well spent to test this using the jar file, as it prompted me to create an overloaded constructor in SearchResultInterface.java that created the graphical user interface that contained a simple TextArea to append the data in the List to (see Figure 3). This means this requirement is met more successfully as it is satisfied whether the program is run in an IDE or straight from the jar file.

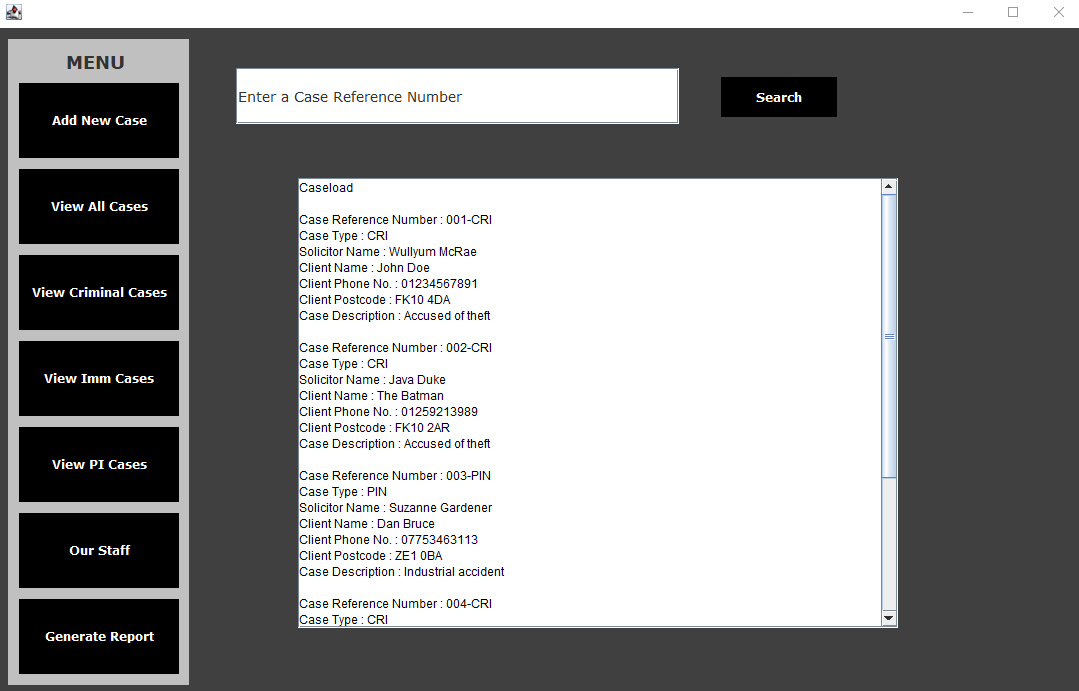


Figure Displaying Cases to screen

What more can be said – the solution displays all cases when the user presses the button, therefore, the requirement has been met successfully.

## Summary

Overall, the solution satisfies the requirements identified at Inception Planning to a large extent. The discussion above highlights what requirements have been fulfilled to a partial extent, with valid justification provided for this decision. A fair amount of testing during Development evidences the fulfilment of the requirements that were focused on in this iteration of development.

# Strengths and Weaknesses

## Technical Documentation

Overall the technical documentation that has been submitted as part of this Graded Unit 2 project has been thorough and provided well‑reasoned justifications for various decisions. I have made effort to provide substantial and well‑informed documentation throughout and various strengths and weaknesses of the technical documentation for each stage of the project is discussed below.

### Inception Planning

This stage of the project involved an Action Plan Report and a Project Plan. The Action Plan Report consists of: an interpretation of the project brief, identification of requirements, analysis and summary of information gathering techniques performed, an indication of the aims of the project, the identification of technical and informational resources; with the Project Plan consisting of commentary on the chosen software development methodology, scheduling of the project, identification of milestones and deliverables, identification of main tasks and sub tasks, and finally a list of resources with supporting explanations. The Inception Planning technical documentation therefore made a solid attempt at providing a thorough plan and analysis of the project.

Its thoroughness is one of the key strengths of the technical documentation. The extensive research resulted in having a lot of background context to draw upon which made the development of the system a little easier in the sense that I knew why a system like this would be necessary. The background research considered a variety of aspects such as job descriptions and similar systems. This meant the Report shows a sense of what the end‑user’s job is like in reality and highlighted factors such as their likely level technical competence. Having this understanding of the end‑user’s technical skill level meant the prototype application didn’t have to be absolutely fool‑proof as the research suggests that Legal Secretaries and Solicitors are likely to have competent IT‑skills.

Similarly, research was also performed on similar systems. This was a very useful part of the technical documentation. It highlighted some common “pain points” of general case management software which guided some of my decisions throughout the design and development of the project. The Inception Planning Report addresses two similar systems. Although this is not a great number of similar systems, it was enough for this project. The two similar systems that were researched were prominent and commonly‑used systems in the legal industry, so I do not think that researching 3 or 4 or 100 similar systems would have made a big difference to my own project. These two similar systems were selected because I had a familiarity with working with them previously, so I knew they were worth my time investigating from a software development point of view. This section of the technical documentation was extensive enough and provided really helpful information in guiding the design of my own project in terms of functionality and accessibility.

As thorough and extensive as the Inception Planning technical documentation is, there are sections that I feel lacked clarity and had potential to cause confusion. The Project Plan includes a Gantt Chart that attempts to show the overall scheduling for each phase of the project. It does show an allowance for slippage time and relevant predecessors are included, but the Gantt Chart is very difficult to read. I won’t hurt your eyes by providing it here, but it can be found at Figure 2 of the Project Plan Report. The Gantt Chart was a necessity because the SQA said so, but it really was not indicative of reality at all. In fact, the Project Plan overall was a bit of a waste of time, and the only reason it was submitted is because the SQA told me to. I am told that in real life, software is developed in teams. But this was an individual project, which means my Project Plan consisted of me myself and I, which ultimately lead to over‑allocated resources:

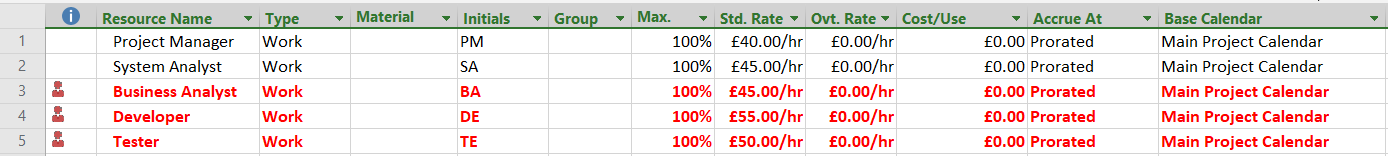


Figure Over-allocated resources

This part of the technical documentation doesn’t offer any real value to the overall project. The only good thing to come of the Project Plan is that it’s shown me I don’t want to be a Project Manager. Ever.

### Solution Planning

Similar to the technical documentation in Inception Planning, the technical documentation submitted for Solution Planning was also thorough and extensive. It attempts to further analyse the problem domain and provides a section dedicated to a use case model, the domain model, a static and dynamic model, along with a view model and a data binding design.

A strength of the technical documentation throughout this stage is its nature of building upon various diagrams. For example, it uses the use case diagram created in Inception Planning, and builds upon that to arrive at a fully‑dressed use case diagram. Similarly, after performing two natural languages analyses, it uses handwritten CRC cards to move towards starting to think about the static model. I think this is a strength of the Solution Planning technical documentation because overall, this stage involves quite a lot of work in a relatively short space of time, so handwriting CRC cards on index cards prevented me from wasting time in getting lost in any software in terms of creating a table (or similar) that represents a CRC card and then making it look nice. The technique of creating CRC cards is an informal technique anyway, and are usually performed using index cards, so I felt it was more appropriate to include it in this format in the technical documentation. It also breaks up the Solution Planning Report a bit, as quite frequently there are walls of text that I think ramble on a bit, so including photos of these handwritten cards can make it a bit less intense for anyone reading the Report.

Soon after this informal technique, the Solution Planning report moves into considering the static model. The CRC cards were a good basis for starting to think about an initial class diagram as they raise questions regarding what can be retained as a software class and what can be binned in terms of software classes. It then offers a balanced discussion of potential class relationships and structures before building upon the initial class diagram. This is a positive aspect of the Solution Planning Report as the diagrams which are produced here raise a lot of questions regarding notation, multiplicity, and package structure. However, this section that details the static model of the software gets quite bogged down in relationship types and notation. This section almost reads like an internal argument with myself and is probably more extensive than it needs to be. In terms of technical documentation, this could be viewed as a downside as it reduces clarity for the reader; however, on a personal level, it was a beneficial process as it allowed me to practise creating class diagrams and to make decisions as to how classes should interact with each other.

Similarly, the use case descriptions provided in the use case model are quite detailed. Considering these are a very useful tool to other developers working on the same project (in the real world), the use case descriptions in the Solution Planning Report could be improved upon greatly if they are broken up more. I don’t think they make use of subflows and alternative/exceptional flows as well as they could do, which, in reality, could make it much more difficult for a programmer to modularise the software. If these use case descriptions were broken up more, they would be of much more value to other team members (if there were any) as they are really there to provide context of the system and to provide detailed requirements of the system. As they’re also used to guide parts of the testing workflow later in the project, they really are useful and important to the whole team working on the project in providing the detailed logic of the system.

Even though the use case descriptions could be made more valuable by being broken down, the Solution Planning Report still makes a fair attempt at developing the dynamic model of the system. It provides sequence diagrams for key use cases: “Log In”, “Create New Case”, and “Search Cases”, and similar to the approach to the class diagrams, it builds upon previous sequence diagrams where appropriate. Accompanying the sequence diagrams are explanations that are listed in singular steps, which I think clears up any ambiguities from the use case descriptions. Following the sequence diagrams are activity diagrams for the same use cases which are kept fairly simple so key decision points and control flows are clear, and are accompanied by explanations that cross‑reference to the use case model appropriately.

In terms of the view model, this section provides appropriate justifications for decision‑making. More specifically, the value of recording minutes of client meetings came through here. Keeping meeting minutes from the onset of the project made it far easier to design the view model, and client meeting number 4 showed the client was happy with this design. Making the effort to do this allowed for the documentation of any clarifications and summaries of the various meetings – if there were a wider team working on a project, this would allow all team members to see client communication and no one would have to rely on memory when considering client feedback.

Overall, the Solution Planning Report includes all necessary diagrams/artefacts to progress with the construction of the system. It has tendencies to become bogged down in too much discussion and analysis, and at times the inclusion of detail actually lessens the usefulness of this technical documentation. However, it does start off with simple diagrams and attempts to build upon them, which is valuable when trying to construct a system that successfully meets its requirements as individual steps are thought through more carefully.

### Development

The technical documentation for the Development phase of Graded Unit is naturally limited compared to the preceding stages. However, there are still a variety of outputs generated in this phase. For example, the Development Report provides explanations of the validation algorithms and shows the process of developing these, leading to the use of regular expressions successfully. It provides supporting evidence of how these regular expressions were arrived at, providing references where appropriate.

The Development Report then transitions into a discussion of program organisation and provides research into the Model, View, Controller organisational technique, which serves as part of my unfamiliar territory for the project. This was a really helpful part of the documentation as it helped greatly in deciding where to contain everything in the system. As this is really the first project where I have had to develop a software system, I was quite confused as to where to keep certain modules of code. But this section of the Development Report really clarified how a program can be structured well and in fact lead into further discussion into password hashing and the exploration of an unfamiliar ethical hacking tool (hashcat).

Furthermore, the Development Report includes sections on developing the user interface, error handling, and internal documentation. It provides evidence of how the graphical user interface was built, including some evaluation of WindowBuilder; some evidence of how error handling was implemented; and evidence of JavaDocs. In hindsight, I do not think these sections add much value to the overall Development Report, but they are useful just for clarity and in acknowledging that these aspects have been implemented in the prototype application. There is also an updated class diagram provided alongside the Development Report which is a nice addition to allow for comparison to the class diagram provided in Solution Planning.

Additionally, there is a variety of Testing documentation provided alongside the Development Report. This accounts for a variety of testing strategies, so the testing workflow covers a wide range of tests that catches different issues. The recording of unit testing (see section The Problem Domain in the Development Report) reveals issues early in Development.

Overall, although technical documentation submitted for Development is limited in comparison to the preceding stages, it is still valuable to the overall project and offers appropriate justifications for certain decisions.

## User Documentation

The Development stage included creating documentation to support the end‑user in using the new application. This was submitted in its own folder, appropriately named “User Pack”:

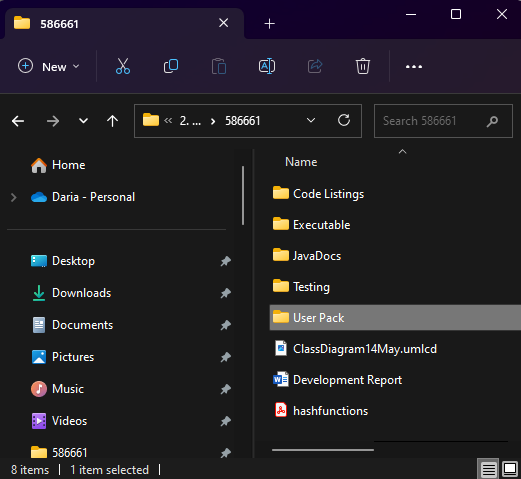


Figure User Pack directory

Inside this directory were three files: a User Guide document, the executable, and a PowerPoint with short clips to demonstrate instructions in the User Guide. This user documentation was used in usability testing, where the tester had no background knowledge of the system. This was a valuable exercise as the tester feedback noted that the User Guide is very clear and offers step by step instructions that are easy to follow. This was the objective of the User Pack – to provide clear guidance on how to use the system. To gain a better spread of feedback, this User Pack could have been distributed to more than one tester, but time restrictions prevented me from doing so as usability testing was conducted towards the very end of Development (the prototype application had to be in a more functional state, which required the bulk of the time permitted in Development). However, asking someone who does not study software development to test the application resulted in genuine and authentic feedback that highlighted where improvements were needed.

Overall, the user documentation achieves its purpose of guiding the user on the usage of the system well by providing clear instructions in the form of written documentation and in the form of video clips.

## Prototype Application

The prototype application was developed with the intention of satisfying specific use cases. This is described earlier in this report under The Solution which also discusses to what extent it satisfied these requirements. The prototype application contains a variety of data validation, which does not allow for bad data to be stored anywhere. The Test Log evidences that the majority of tests pass, and the few that fail are rectified with explanations provided towards the end of the Test Log. The prototype application could make use of better worded error messages. This was noted in the feedback from usability testing, with particular reference made to the error message displayed when an invalid phone number is entered. It was indicated that this error message does not inform the user about the length requirements of the data, so if the number entered is too short, there is nothing in the prototype application that communicates this to the user.

In addition to implementing data validation, the prototype successfully implements a strong password hashing algorithm. Given the background context of the project brief, developing a system that is secure is a priority, so the extra time spent on trying to implement the BCrypt password hashing algorithm was time well spent as this means an attacker will find it much more difficult to crack the users’ passwords. This was a deliberate choice of password hashing function as one other hashing algorithm was explored – SHA‑512, which Java supports the use of. Although this was easier to implement and required less code, BCrypt is by far the stronger algorithm, so the program having the ability to utilise this algorithm is one success of the prototype application.

The application was developed as a graphical user interface, which in itself has a variety of pros and cons. One of the “pain points” identified in the investigation of similar systems was the difficulty in locating certain elements of the system as a result of not implementing a high contrast (see page 34 of Inception Planning Report). The prototype application, on the other hand, used a colour scheme that has a much better contrast ratio. The colour scheme was run through two contrast checking tools during Solution Planning (see Figures 44 and 45 in Solution Planning Report), so overall the prototype application promotes ease of use by making it easier for users to read information.

In contrast, the graphical user interface isn’t really exciting or striking. It is a simple GUI with a vertical navigational menu on the left hand side. There isn’t really much of a balance between ease of navigation and pleasing aesthetics. It is clear how having a professional designer such as a graphic designer would help significantly in software development. The GUI of the prototype application uses plain and boring colours:

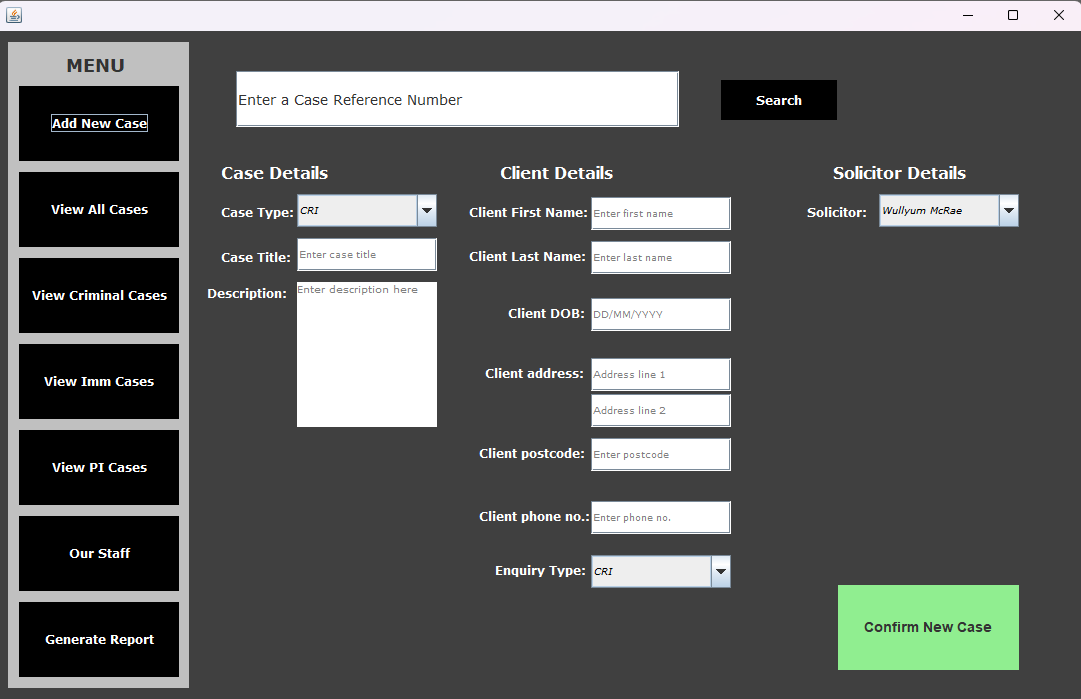


Figure Graphical user interface

Although designing a striking GUI wasn’t a primary aim of the overall project, it looks really plain and bland. This isn’t a major flaw of the prototype application though: given the background context of the project brief, management systems are generally not that nice looking.

The prototype application can also be quite difficult to navigate at times. For example, in the interface shown at Figure 6, the user is not able to tab down the way. This would probably result in some user frustration, however given the timeframe of the Development phase I did not want to waste hours researching how to program this in Java Swing. This is made worse by having all data input for adding a new Case in the one window. This was a deliberate choice as I took into consideration that the client might have the need to add many new Cases to the system at once, so it seemed easier for them to enter all the details of a single Case at once, and then clear all the fields ready for the next Case to be added.

Furthermore, there is a potential “pain point” in the prototype application. The investigation of similar systems notes (at page 37 of Inception Planning Report) that in LawWare, there is no obvious sort functionality. Similarly, if the user wishes to view all Cases in the prototype application, they cannot specify how they would like the Caseload to be ordered. This doesn’t impact the overall functionality of the system greatly, it simply means the Cases are displayed in the way they sit in the data structure, but they are still displayed in a structured manner nonetheless.

In terms of the data that the system receives, the prototype application currently does not implement data persistence. This would be a necessity in the real world, but for this project the prototype application focuses on data security and integrity first and foremost.

Overall, the technical documentation relating to Development is balanced and offers extra explanation where appropriate, particularly in terms of data validation, code organisation, and data security.

# Future Development Recommendations

## Data Persistence

In future development of the solution, it would be hugely beneficial to implement **data persistence**. Although the Inception Planning Report highlighted various risks associated with a Relational Database Management System, it is still an effective way of storing data and would allow for the quick and easy retrieval of data. A RDMS supports all the crucial CRUD functions and undoubtedly has the capacity to cope with growth. Not only is it a time and cost‑effective way of implementing data persistence, but it provides a centralised and structured storage of data where data integrity rules ensure correct and in a consistent state. There are a few reasons for the prototype application not implementing data persistence, such as a lack of experience on my part and a desire to focus on satisfying key use cases well and to a high standard. But for future development, this would be a strong recommendation as the application isn’t wholly useful without data persistence.

## Sort Functionality

Another recommendation that would make the application more useful is the functionality of **sorting data**. The prototype application does not offer this functionality because it didn’t fall within the scope of the use cases that were focused on. The investigation of similar systems carried out in Inception Planning makes reference to the user’s ability to sort matters (Cases) by date or by matter name. This is certainly something that would enhance the user experience as it would be much easier for the user to filter out Cases that are not relevant to them. The investigation of similar systems in Inception Planning also highlighted that iManage allows the user to **create a “Favourites” list** so that the user can access Cases that are very active. The ability to perform various sorting functionality would therefore make the user experience much smoother and is recommended for future development, though not as valuable as implementing data persistence.

## Document Management

In addition, it would also enhance the user experience if the application allowed for **storage of other Case details**. For example, similar systems (specifically iManage) allow the user to store legal documentation which the user can easily navigate to after searching for a Case. Other Case details appear in a tree‑like structure on the left side of the screen to allow the user to easily navigate to whatever section they need. If the user searches for a Case, the Case appears as a directory icon as the “root” of the tree. This can be expanded to reveal sub‑directories where the user can access legal documentation, various correspondence pertaining to the given matter (Case). I would recommend this for future development because it really enhances the concept of managing a legal firm’s caseload; this wasn’t implemented in the prototype application because it would have caused a serious case of scope creep. It’s unlikely that a legal firm would want a software to be developed that only allows for adding Cases and viewing Cases – this is certainly a good starting point, but to gain any real business value, much more functionality is needed and the centralised storage of *everything* relating to a given Case (documentation, correspondence with client, correspondence with opponent etc) is what would bring true value to the business.

## Invoicing

Furthermore, there are recommendations that could enhance the application after the use case “Record Billable Activities” is implemented. The purpose of this use case is for the business to keep track of monies owed to the firm. To make the overall workflow smoother, a helpful feature could be to implement an **invoice generator**. Similar to the This would add business value to the application as it makes the application more centralised in the sense that duties relating to case management can be performed from within the one software.

## Summary

In summary, there are a range of recommendations which, if implemented, would really increase the business value of the software. These are summarised in the table below.

| **Recommendation** | **Justification** |
| --- | --- |
| Data Persistence | * Key to all other recommendations * Prototype not wholly useful if data not permanently stored |
| Sort Functionality | * Enhances user experience * Easier to view the Cases the user needs to view |
| Document Management | * Cases generate a lot of documentation * Integral to concept of case management * Also enhances user experience |
| Invoicing | * Would make workflow smoother and more efficient * Reduces employee labour spent on creating invoices; frees their time for other tasks * Centralises case management tasks (everything in one place) |

# Modifications and Unforeseen Events

The project on the whole has not seen a huge number of modifications being made. There was the addition of a new use case after client meeting number 3 (see Solution Planning Report), “Add Staff Member”. However, this did not impact the project much at all as the use cases which were the focus of the Development stage were satisfied without implementing the new use case.

## The Static Model

There were a number of refinements undertaken when the static model was being built. The initial class diagram started with basic classes, but the second class diagram required a number of modifications with regard to relationships. The final class diagram presented in the Solution Planning Report reflects these modifications, but there is acknowledgement that it is subject to change. At the end of the Development phase, an updated class diagram was submitted in PDF format for the purpose of comparison with the class diagram provided in Solution Planning.

There are certainly similarities between these two diagrams. For example the classes Client and Employee still inherit from the class Person. However, there is the addition of several Controller classes and View classes which are vital to the success of the system but are not incorporated into the class diagram provided in Solution Planning. I wouldn’t say the class diagram in Solution Planning was flawed – it captured a lot of important details and relevant relationships that **contribute** to the success of the application, but the modified class diagram provided in Development shows the addition of classes which, if they were omitted, the application would not achieve its requirements at all. For example, FR1 requires log in details to be validated. To achieve this functional requirement, the application needs the class LogInController, shown in the modified class diagram. The Solution Planning class diagram accounts for an Employee object to have a username and password value, but the class only shows standard getters and setters for this data; it does not indicate that that there is any kind of validation routine to successfully validate the user’s log in details. In contrast, the updated class diagram has a Controller class dedicated to implementing log in validation:

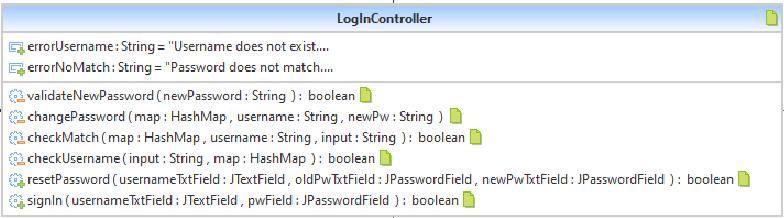


Figure Updated class diagram snippet

The same can be said for FR2 and its sub‑requirements. The Solution Planning class diagram captures Case details, but it doesn’t indicate any validation routines. The updated class diagram contains a class NewCaseController, which contains methods to receive only valid data. To achieve certain requirements to better standard, these modifications to the class diagrams were necessary and they ensure that the system will only allow the user to enter good data.

## Wullie

It took some time to implement these changes and decide where to include the methods that acted as validation routines. Thankfully, work on the development of the prototype application wasn’t left until the last minute, so these necessary modifications didn’t really hamper the success of the prototype application. Starting work early was actually a key factor to managing to submit assignments and keep up with coursework as far as possible. An unexpected event came in the form of Mr Wullie McRae dropping a bombshell on us all by announcing his departure from the College. This resulted in a deadline being brought forward quite considerably. I hadn’t made a huge amount of progress with the room booking assignment at that point, so focus was taken away from Graded Unit to spend time on the room booking assignment. Fortunately, Graded Unit Development was started early and given that I’d spent time on it during the Easter holiday, it didn’t have a catastrophic impact on Graded Unit.

## Stirling University

Another unforeseen event came in the form of a time management crisis. I have also made reference to the half‑module undertaken at Stirling University in the section Personal Reflection, but it is applicable here also. I absolutely did not anticipate this half‑module to take significantly more time than what was expected.

This was, from my perspective, largely owing to a lack of academic support from Stirling University. An element of independent learning is always expected in university education, but the teaching of this half‑module consisted mostly of short, online video clips. The practical sessions mainly consisted of confirming your attendance, and any attempt to ask for help resulted in engagement with the lecturer that typically lasted no more than about 20 seconds. This meant that many, many hours were taken away from Graded Unit and as such I had to use most of the February break and more to try and navigate my way through the assignment for Stirling University. I sourced as many relevant academic sources as I could, tried my absolute best to ask questions in practical sessions, and watched the (brief) video clips that were posted online to try and complete the assignment. The juggling of this at the same time as Graded Unit was all consuming, but some attempt at dividing my efforts paid off somewhat as the Solution Planning submission achieved as many marks as it could have done. I don’t know the grade given to my university assignment, and I’m not sure if we will find this out owing to a university marking boycott, so I can’t comment on whether I learned what I was supposed to learn (it doesn’t feel like I have). This half‑module and Wullie leaving the College were the most notable unforeseen events.

## Other Modifications

Aside from these two unforeseen events, a few other, minor modifications were made to the prototype application as it was being developed. Firstly, the link in the Log In window changed from “New Employee? Click here” to “Click here to set or change your password”. The client was taken shown this modification in client meeting number 5 (see Development Report) and was agreeable to implementing this change. I felt this modification was necessary because it was clarified that the functionality of resetting password should be made available to any Employee; not just a new Employee.

This change also resulted in some tweaks to error messages that were to be displayed. The resetting of a password wasn't a clearly identified requirement in Inception Planning, so if this hadn't been implemented it might've freed up time to implement other use cases. However, it's an important functionality as far as security is concerned, so after the client provided confirmation that they would like this functionality implemented I decided to change the error messages to indicate whether the username existed or not, and whether the password matched or not. I kept in mind that in reality you probably wouldn't display this information as it could be used tactically by a malicious actor, but in the prototype application I wanted to show that there are checks in place to ensure the username is evaluated and that the password is evaluated. This additional functionality took some extra time to implement, but it was worth implementing because a) it makes the system more secure by allowing the user to set their own password and therefore no hard‑coded values can be used by an attacker (hash values only were stored); and b) it's good experience for me to devise the different steps and decision points involved in the log in routine from a programming perspective.

Another minor modification made in the prototype application concerned the format of output of Caseload data to screen. The design of the graphical user interface for the use case "View Caseload" provided in Solution Planning is shown below:

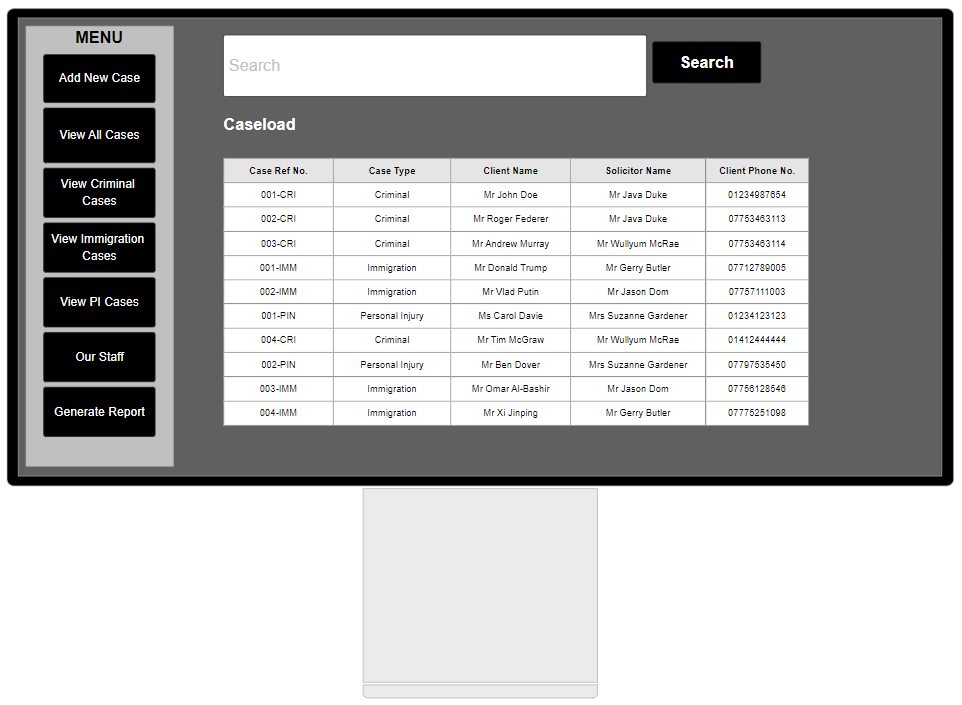


Figure View Caseload GUI design

I have already acknowledged that Functional Requirement 5 was only partially implemented. This was primarily for testing purposes, so rather than spending hours farting around with Java Swing trying to present the data exactly as above, I changed the format of the output so the Caseload data can still be viewed, just in a slightly different manner. It is definitely a modification of Figure 8 above, but it is still displayed in a structured way which I would argue is actually easier to read (see Figure 3 and also Figure 9 below for ease of reference):

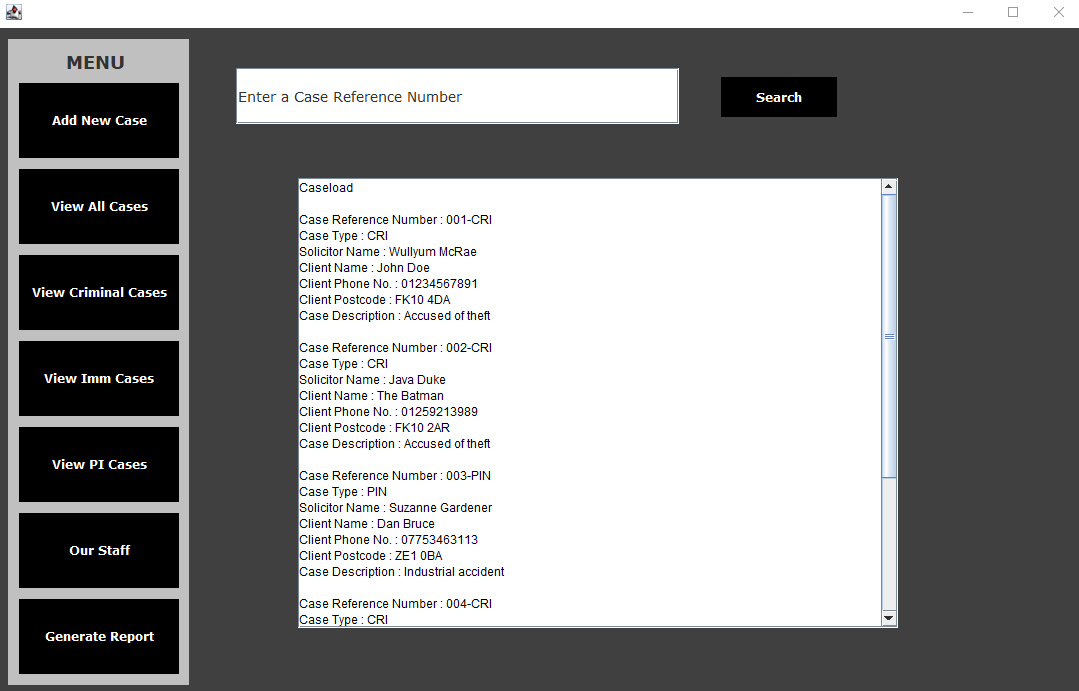


Figure View Caseload modified GUI

As mentioned, this saved a lot of time farting around with Java Swing, so approaching this use case in this way actually helped the development process and aided testing workflows throughout.

| **Modification/Event** | **Comments** |
| --- | --- |
| New Use Case | * Minimal impact – other use cases were prioritised |
| Class diagrams | * Noticeable addition of detail in most recent class diagram * Vital to success of system |
| Wullie left | * Moderate impact – other deadline brought forward; took time away from Development * Impact minimised as Development work began early |
| Stirling University | * Time management crisis * Felt all‑consuming; deadline very close to College deadlines * Resulted in rushing some areas |
| Other | * Password reset functionality – enhances security * Log In link tweaked to more appropriate message * Error messages better inform user what's wrong * View Caseload – change to format of data output |

# Personal Reflection

The overall project assignment has brought a mixture of challenges and achievements. It’s allowed for the development of both technical skills and knowledge alongside various meta skills that are not specific to software development but complement software development well.

## Programming

A lot of effort has been spent on developing technical skills and knowledge throughout the project overall. From the outset, I made a conscious decision to concentrate time and energy on the practical side of this project, specifically Development, solely because I wanted to make an attempt to develop confidence and self‑belief in programming. As for the course overall, learning to program with no prior experience has been a major learning curve. This made the Development stage of the project assignment a daunting prospect, as until this point there’s generally been ample support provided where programming is concerned.

However, with this being the first experience of developing something you can call a functional application, this project has given me an excellent opportunity to really practise programming skills. It’s been really nice to realise that programming skills aren’t limited to those who are mathematically gifted, and having the freedom to approach the Development stage independently allowed me to choose where to start and how to piece everything together. The exploration of the organisational technique Model, View, Controller was really worthwhile in learning how applications are commonly structured. Indeed this has probably not been implemented perfectly, but I’m pleased that I managed to abide by this organisational technique to an extent that lead to the satisfaction of key functional requirements.

## Ethical Hacking Tool

Furthermore, exploring some new ethical hacking tools was another learning curve. The purpose of doing this was not to satisfy the SQA’s requirement of using an unfamiliar library/construct, but to give myself some exposure to downloading and installing a virtual machine with a new operating system and then using the command line to execute instructions. Although Kali Linux is not completely new to me since it was used in our Ethical Hacking unit in 1st year, we were usually always given guidance and instructions on any class exercise. On reflection, I felt I never really understood what and why we were doing something, and considering the title of my degree is Software Development with Cyber Security, it seemed fitting to explore security in this way a bit further. For example, to download Kali Linux in the Ethical Hacking unit, a direct link was provided to us, and the application to set up a virtual machine was already installed on College machines. So the process of just downloading VirtualBox was a learning curve in itself. But considering I could just about turn the computer on in 1st year, I’m pleased I’ve managed to go the extra mile here in setting up the virtual machine with Kali Linux and then execute a simple password cracking attack. The attack wasn’t successful, but I took that as a positive as the other purpose of this was to test the security of the password hashing algorithm used. Overall, this exposure to something that seemed like second nature to other students last year has built a bit of self‑confidence.

## Time to Face the Ternary Operator

In addition, I’ve tried not to shy away from things that I’ve not felt comfortable with. For example, the ternary operator has always flustered me and I’m sure that’s something real developers won’t think twice about using. Throughout constructing the prototype application, I made a conscious effort to try and use the ternary operator a few times. Specifically, it’s used a variety of data validation successfully and I feel more comfortable about using the ternary operator now. It probably seems like an unimportant thing, but again, no coding experience in August 2021 to using shorthand Java = a big step for me.

## Meta-Skills

However, the skills developed through this project assignment are not solely technical. A variety of meta‑skills have come into play through the project. These skills are described by Skills Development Scotland (SDS) as “innate, timeless, higher‑order skills that create adaptive learners and promote success in whatever context” (Meta-skills Progression Framework Self-management Social intelligence Innovation, n.d.). They fall into three categories: self‑management, social intelligence, and innovation. Personally, I feel the meta‑skills that have been developed the most throughout this project are adaptability and focus (both categorised as **self‑management**), along with critical thinking and sense‑making (both categorised as **innovation**).

### Self‑Management

As an individual who is not naturally adaptive, this project has required me to exercise adaptability far more frequently than I’m used to. I didn’t know what to expect with Graded Unit, and the workload that it generates combined with managing other units in the year has been more difficult than expected. But managing to submit everything on time and achieving a healthy score for other coursework along with each stage of Graded Unit so far has undoubtedly required an element of adaptability and focus. SDS refer to resilience as part of adapting, and I didn’t believe I had this level of resiliency at the outset of the project.

In their Meta‑Skills Progression Framework, SDS also refer to adaptability as “flexibility when handling the unexpected, adapting to circumstances as they arise”. I without a doubt would not have described myself as flexible “when handling the unexpected” at the start of this project. However, throughout the project there have been a couple of unforeseen events where adaptability has been a necessity. At the same time as the Solution Planning stage of Graded Unit, a half‑module undertaken at Stirling University took significantly more time than expected and also caused significantly more stress and negative energy than expected. This was in part due to the lack of support and engagement from Stirling University: I fully anticipated a greater degree of independent learning as this is typical of university education, but this half‑module entailed virtually no teaching at all with the exception of a few short, online video clips. It really sucked out a lot of passion for learning and provoked a lot of self‑doubt for me. It took many hours away from Graded Unit, which only created more stress and anxiety. However, despite this, the work submitted to Stirling University was still not a half‑arsed effort. I made an attempt at it and also managed to submit Solution Planning documentation on time that attained the full possible mark distribution for this stage.

Reflecting on the project, the period of time between starting Stirling University’s half‑module and submitting Solution Planning is definitely where time management could have been improved. This period of time really felt all‑consuming and it’s highlighted my poor perspective of time; a better balance really could have been struck which I think would have allowed me a better learning experience as far as Solution Planning is concerned. The poor time management at this point resulted in some sections of Solution Planning being more rushed than normal, which, for me, doesn’t allow for learning as effectively and is definitely where improvements could have been made. Thankfully, the university assignment and Solution Planning weren’t left until the last minute and both were started as early as possible, so at least I had some sense in that regard.

### Innovation

The final meta‑skill I feel should be discussed is critical thinking. This skill can include, according to SDS, deconstructing, logical thinking, judgement, and computational thinking. Until it came to evaluating this Graded Unit project assignment, I didn’t realise that critical thinking encompassed such abilities. The skill of critical thinking could have made the project better in terms of, for example, breaking down use case descriptions more – which was noted in Solution Planning feedback notes. If I were more conscious of critical thinking, I think I would have produced use case descriptions that reflected more modular programming. However, I also think there are other areas of the project where I’ve really demonstrated an ability to break down the overall system into smaller pieces, specifically throughout Development. The Meta Skills Progression Framework describes that a learner shows “Senior Phase” level critical thinking in “explaining complex processes or concepts with relevant supporting detail and evidence and can confidently justify their reasoning” (Meta-skills Progression Framework Self-management Social intelligence Innovation, n.d.). The feedback provided for Solution Planning makes reference to extensive detail provided and a skill to justify my decision making. This sentiment is reiterated in reference to class diagrams, noting that “extensive justification of…potential classes is very good and the progression as you determine your system is valuable” (Solution Planning Lecturer Feedback, May 2023).

There is obviously good evidence of deconstruction of problems in the project assignment, but I also think critical thinking could have made the overall process better not only in terms of breaking use case descriptions down, but also in terms of not becoming bogged down in very small details. The meta‑skill sense‑making includes an ability to think holistically and had there been a better awareness of sense‑making throughout the project assignment, I think more requirements would have been fulfilled. This is not to say sense‑making was an under‑utilised skill in the project; there is still evidence of using informational resources to make well‑informed decisions, for example, but the deconstruction of the problem resulted in me focusing too much on smaller details (like data validation routines) that I often forgot about the bigger picture.

Overall, there’s undoubtedly been a variety of knowledge and skills gained from this project assignment, but equally there is plenty of room for improvement. The biggest success for me personally has been seeing the progress from starting this course in August 2021 and not knowing what an IDE is (is that where you type the code in, is what I used to ask) to abstracting code that hashes a password using an algorithm I absolutely cannot understand is a big jump. The Graded Unit project has brought challenges and achievements alike, but on the whole I think it’s increased my self‑belief that I’ve made the right choice in choosing to study this course.

# Bibliography

Meta-skills Progression Framework Self-management Social intelligence Innovation. (n.d.). Available at: https://www.skillsdevelopmentscotland.co.uk/media/48745/meta-skills-progression-framework-final.pdf.