# Developing a web application to facilitate Appointment management and communication among support staff.

Contents

[Developing a web application to facilitate Appointment management and communication among support staff. 1](#_Toc19455282)

[Problem description 1](#_Toc19455283)

[nature and context of the problem 2](#_Toc19455284)

[analysis of likely impact 3](#_Toc19455285)

[An account of related literature 5](#_Toc19455286)

[The account of project work and its outcome 8](#_Toc19455287)

[Part 1 – Inception, Analysis, and initial design. 8](#_Toc19455288)

[Part 2 - underlying object representations and appointments. 15](#_Toc19455289)

[Part 3 - website design and restful links to Java backend 17](#_Toc19455290)

[Part 4 - website design and note system. 36](#_Toc19455291)

[Part 5 - admin dashboard and advisory sub-system. 44](#_Toc19455292)

[Part 6 - behavioural flagging and advisory sub-system. 55](#_Toc19455293)

[Review of current stage of project work 58](#_Toc19455294)

[Review project management 58](#_Toc19455295)

[Resources, skills and methods 58](#_Toc19455296)

[Review of personal development 61](#_Toc19455297)

[References 63](#_Toc19455298)

[appendix A 1](#_Toc19455299)

[appendix B 1](#_Toc19455439)

## Problem description

A charity runs a community based supported living project in the UK. They plan to replace their current paper-based scheduling and management system with a website.

This new system will allow managers to add or remove clients, support staff members, and supported houses. To set up timetables and access appointment meeting notes. The manager will also have the facilities to create and change client support teams.

Support staff members will be able to access their timetables. Their timetables will display which client they are to see and what time the appointment is to start. They will be able to record when they started the support session, the support session ended, and what was done. Support staff members will also be able to send and receive messages related to a client they support and specify the importance of the message.

The system will handle potential problems like a client being assigned two or more key workers and timetable conflicts based around travel time and appointment availability between a client and support staff member. The system will also flag several set behavioural patterns, e.g., a client is continually cancelling support or a client withdrawing from the assistance of a specific support staff member.

### nature and context of the problem

* What is the problem that needs to be solved?

A reliance on using written documents to manage and record staff appointments and the need to use text messages and phone calls for passing client information between multiple support workers.

* Why is it considered a problem?

Each support worker holds an extensive collection of paperwork related to each person they support this takes up support hours and means much time is spent checking paperwork from different support workers associated with a single client this can mean behavioural signs are missed.

Using phone calls to liaise between multiple support workers is difficult as a support worker may not be able to answer the phone in a support session this can have severe repercussions in the response time to a client in distress.

* What will be the benefits of solving it?

Fewer support hours will be spent on paperwork. Introducing a priority-based messaging system will improve communication among support staff. The missing of messages being one of the most common complaints from support staff.

Also, a system to automatically flag common behavioural signs and inform the specific client’s key worker ensuring these behavioural signs are not missed; this is one of the concerns for key workers that manage multiple clients with the most commonly missed behavioural sign being cancelled appointments. These will all help in the task of supporting clients.

* What are the key ICT aspects of this problem?

To replace the current paper-based system with a website accessible from any device with a browser, this will allow access to and modification of information about appointments, appointment notes, clients, support staff members, and the charity-controlled houses that act as clients accommodation while at the charity.

The website will have the functionality to automatic handle some behavioural checks, message passing, retrieving information, or saving data to a database. Another problem will be the need for the system to be useable by support staff on the move. As most support staff will use the system moving between appointments.

* What is your existing knowledge of this problem?

I already possess the Java knowledge to create back-end object representations. Also, I have used some HTML and CSS before. The areas I will need to learn related to the back-end database design and on the Front-end, JavaScript, and JSON to update the web page. Restful API to pass JSON objects to and from the back-end Java. Also, linking the back-end java to the database through SQL.

* What might the solution look like?

The solution will be a web app that runs through the device’s browser in the style of the charity’s homepage. The web app will implement responsive design so that functionality and layout will fit the screen size of the display. The plan will be different depending on who has been authenticated a manager or support staff member. A manager will most likely access the web app from a desktop PC and so have more screen space available where support staff members will be accessing the web app from a mobile device on the move.

The information displayed for a specific support staff member will be their appointments for that day, along with prioritised messages. The website will also have the facilities to access past appointment records, display information relating to a client, support worker, or the houses managed by the charity that act as client accommodation.

* What, specifically, will you deliver by way of a project output?

A working prototype of the web application and documentation of the development process. A version of the Volere template and artefacts created during each increment’s analysis and design phase. e.g., class diagrams, object diagrams, and sequence diagrams where appropriate.

A working prototype of the web application will allow the stakeholders to give a final evaluation of the system. Also, documentation will enable them to continue the development of the system or its integration.

* The scope of the system

The scope of the system will be limited to Appointment management, behavioural monitoring, and messaging. Anything outside of that core set of functionalities will be added to a waiting room of possible expansions to the system later. The waiting room list is in appendix A page 60

### analysis of likely impact

The likely impact of this piece of software firstly relates to if it meets the expectations of the stakeholders, their requirements. Secondly, even if it does meet the requirements of the stakeholders, how well will the charities support staff, clients and managers take to the new system.

Some of these concerns are part of the development process, trying to incorporate staff into the development decisions to create a sense of ownership early on.

How clients take to the new system will be more complicated as under the current system, a client is given a timetable paper each week. Replacing this with a digital version may not always work; not all clients will have a phone, access to a computer or internet. So, in some cases, the new system will need to act like the old one concerning clients without access to the necessary equipment.

However, if the new system meets the needs of the charity and is accepted by the staff and clients. Replacing an in-use paper system would be a challenge and probably require implementing the system in several test cases with trained staff and managers and a small selection of clients used alongside the paper system to at least keep the same level of care.

Another area of impact will be the cost of not just implementing the new system but its maintenance. Using a software-based system will always be more expensive to maintain than a paper-based model, but there is the possibility of the system overall be more cost-effective relating to the working hour costs. If managers and support staff can work more effectively with the new system, it could save time on paperwork and communication. When the project is complete, and under test, this could be checked how long a manager takes to set up a new client using the paper system or software system.

One of the leading legal and professional issues with this project is the storage and handling of sensitive data — the information relating to a client’s mental health.

Before a client can receive care, they or a representative sign a permission form relating to their personal information; this allows medical disclosure to their direct support team and the support team managers. If the client or representative do not sign this permission form, then the charity will not be able to provide support.

So, the system should only allow authenticated support staff members access to client data they are authorised for, Information about clients they directly support. Considering the release of sensitive data is a crime under the GDPR; this is a paramount concern for the charity.

The charity will need to consider how it will hold this data, directly in their servers or a third-party cloud provider. If the information is to be hosted by a cloud provider, then how this relates to the laws regarding data protection will need to be examined.

When discussing data protection, there are two essential terms to understand. ‘Data Controller’ and ‘data processor.’ These terms relate to who is responsible for personal data under the GDPR.

If the charity were to move its data to a cloud service, that data would not be held in isolation. It would be part of shared infrastructure separated virtually. If at any point, there was a data leak. The charity as the data controller would be liable and not the cloud provider as they would be considered the data processor.

Another important consideration is where the data will be stored. This question of jurisdiction is complex. Moreover, could mean that the charity as a Data Controller could be required to release data legally to a different country and in doing so, be in violation of the GDPR; this can be managed by knowing and having control of where the cloud provider is storing the charity data. For example, storing the charity’s data in the European Union will mean it follows GDPR as all EU member states enforce the GDPR.

All of this needs a more detailed investigation into the charity’s responsibilities based on their decisions regarding the processing and storage of sensitive data. Even if the charity decided to store that data themselves in their servers, the data would need to be secured at rest, through some form of encryption and secured on the move using TLS/SSL.

## An account of related literature

**Suzanne and James Robertson (2013)** **Mastering the requirements process: getting requirements right. 3rd ed.**

Comments: A book on the processes in creating a ‘correct’ requirements document uses a more detailed version of the same Volere template taught by the OU in TM354.

Suzanne and James Robertson (2013) outline a requirement process that can be tailored to the project that is under consideration. From initial problem modelling to a completed requirements document. Each section relates to a part of the requirements document, and it started with a ‘Formality Guide’ with Rabbit projects more related to an agile method. Horse projects with more emphasis on requirements elicitation than an agile method. And Elephant projects where requirements need to be specific and correct. For projects around safety-critical systems.

Suzanne and James Robertson put the argument for the involvement of stakeholders in all elements of requirement elicitation in the forefront. The case for the creation of several artefacts is around stakeholder inclusion and feedback this removes some of the rigidness from the requirements process and hopefully returns a document that is a solution to the problem.

For my project, I followed the Volere template closely. I am treating my project as a rabbit project: less documentation and artefact creation. The only place I deviated from the model was using the TM354 software functional requirements. This choice was based on my own familiarity with the TM354 process over the Volere process.

Overall, I found Mastering the requirements process supported me in producing a requirements document that I feel will allow me to create a software system to meet the charity’s needs.

Mastering the requirements process is Cited 1968 times by google scholar and is well regarded as a starting point for learning software development.

Scholar Google, ‘Mastering the requirements process’. Available at <https://scholar.google.co.uk/scholar?hl=en&as_sdt=0%2C5&q=Mastering+the+requirements+process+&btnG=> (accessed 15 March 2019)

**Walls, Craig (2014) Spring in Action. Fourth edition.**

Comments: A book relating to the technical elements of Spring Framework. While I found this book useful for getting a basic understanding of how Spring framework operated, it did not provide the level of detail I required. An example of this while the book covered database integration it did so using JDBC and the information was somewhat out of date from what I read from the Spring framework official documentation.

Spring in Action did provide a basic overview, but I found myself making use of other information sources, namely the Spring Framework documentation and a website called baeldung. Cited by 561

Scholar Google, ‘Spring in Action’. Available at <https://scholar.google.co.uk/scholar?hl=en&as_sdt=0%2C5&q=Spring+in+Action&btnG=> (accessed 8 September 2019)

**Bloch, J. (2017) Effective Java. 3st edn. Addison-Wesley Professional.**

Comments: a well-regarded book on creating useful code has sections on serialisation, concurrency, exceptions, and lambdas.

One area I used effectively from this book was to use custom error messages along-side Spring Framework to send an error message along with an HTML error code this allowed my front end javascript to display that message from the back-end to the user.

Another area used from this book was to make the staff class abstract and at runtime choose if a class needs to be a support staff or manager with a simple boolean.

Overall this is a beneficial resource to create more effective code and something I see myself rereading as I take on new projects.

Cited 957 times.

Scholar Google, ‘Effective Java’. Available at <https://scholar.google.co.uk/scholar?hl=en&as_sdt=0%2C5&q=Effective+Java&btnG=> (accessed 15 March 2019)

**Head First Design Patterns. 2st edn (2014). O'Reilly Media.**

Comments: I have read other headfirst books and find them excellent for understanding new ideas this goes into detail of several design patterns that were touched on in TM354 this will be used alongside effective Java for creating robust code. Head First Design Patterns is Cited 827 times and is commonly used for learning design patterns with java.

Scholar Google, ‘Head First Design Patterns’. Available at <https://scholar.google.co.uk/scholar?hl=en&as_sdt=0%2C5&q=Head+First+Design+Patterns&btnG=> (accessed 20 March 2019)

**Garcia, A. and Farcic, V. (2018)** **Test-Driven Java Development - Second Edition. 2nd edn. Packt Publishing.**

Comments: an overview of incorporating TDD into a project. That I plan to use for regression testing between increments, the chapters on unit-testing, user-interface testing will be particularly useful.

Garcia, A. and Farcic, V. discusses how to implement the test-driven approach to development this relates to designing and writing the test before the code and using both Black-box and white-box testing. For my project, I focused on white-box testing and unit testing — chapter four of the book.

Using Garcia, A. and Farcic, V method of writing tests before the code and then writing the code necessary to pass those tests. After the functionality is complete, refactoring the code to make improvements without changing the external behaviour and rerunning the test.

Using this method and junit4 with java asserts, I was able to implement a level of regression testing into my project as well. After every code change, all unit tests could be rerun.

Three weaknesses of the TDD approach were Integration testing, acceptance testing, and UI testing. These are discussed by Garcia, A. and Farcic, V, but more detailed research will be needed.

Overall, I found that Test-Driven Java Development helped me create and use unit testing within my project. And have already seen that JUnit tests have improved debugging and regression testing of new functionality.

Test-Driven Java Development is not a heavily cited or reviewed book. So by no means an authoritative discussion on TDD. However, the chapter I used helped me achieve my goals with this book.

**Larman, C. (2005) Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development /. 3rd ed. Upper Saddle River, N.J.: Prentice Hall PTR, c2005.**

Larman, C. (2005). Takes an object-oriented approach to the analysis and Design phase this is done through a case study with each section of the book relating to a part of its iteration.

There is some overlap with Suzanne and James Robertson (2013) Mastering the requirements process. Regarding the use of use cases.

I found Larman, C. (2005), use case models to be more in-depth with a better explanation of extensions or alternative flows. I believe this would have allowed a better-realised use case as I would have explored the if flows in a more structured way.

Larman, C. (2005) also has several chapters on General Responsibility Assignment Software Patterns (GRASP) this helped me with the class models and sequence diagrams to attempt to keep the GRASP patterns of information expert and creator in mind.

Larman, C. (2005) chapters on UML sequence diagrams, class diagrams, and object diagrams contain information very similar to that taught by TM354 software engineering. And I created several of these diagrams moving from my requirements document through the analysis and design phases.

I found Applying UML and patterns to be an excellent book that helps me feel confident taking my requirements document and start coding my system from the UML diagrams and operation Contracts I created from this book.

This book was also highly recommended when I was looking for a book on analysis and design. It is cited 3877 times on google scholar and reference by TM354.

Scholar Google, ‘Applying UML and patterns. Available at <https://scholar.google.co.uk/scholar?hl=en&as_sdt=0%2C5&q=Applying+UML+and+Patterns&btnG=> (accessed 20 March 2019)

**The Open University (2013),’ Resources’, TM354 – Blocks 1,2 and 3.**

Comments: the blocks from TM354 provide a detailed review of the processes of software engineering. All of these will be useful at different stages of development. Particularly the case study of a fictional company producing a software system.

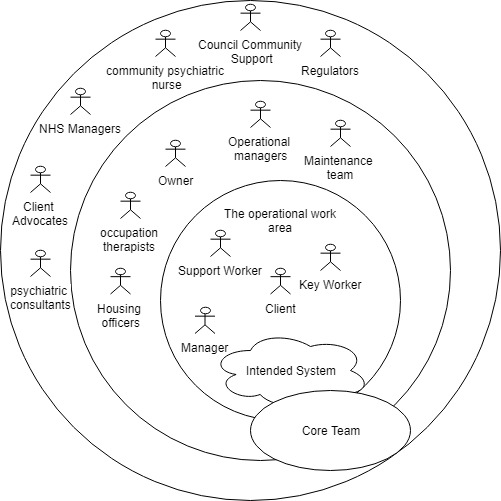
## The account of project work and its outcome

### Part 1 – Inception, Analysis, and initial design.

During the conception of the project, it was decided the best option to understand and create a software solution for the charity was to use the Volere template and take an incremental and iterative approach through researching the Volere template two information sources were chosen as references to its implementation. The first being ‘Suzanne and James Robertson (2013) Mastering the requirements process’ and UML modelling was Larman, C. (2005) Applying UML and patterns.

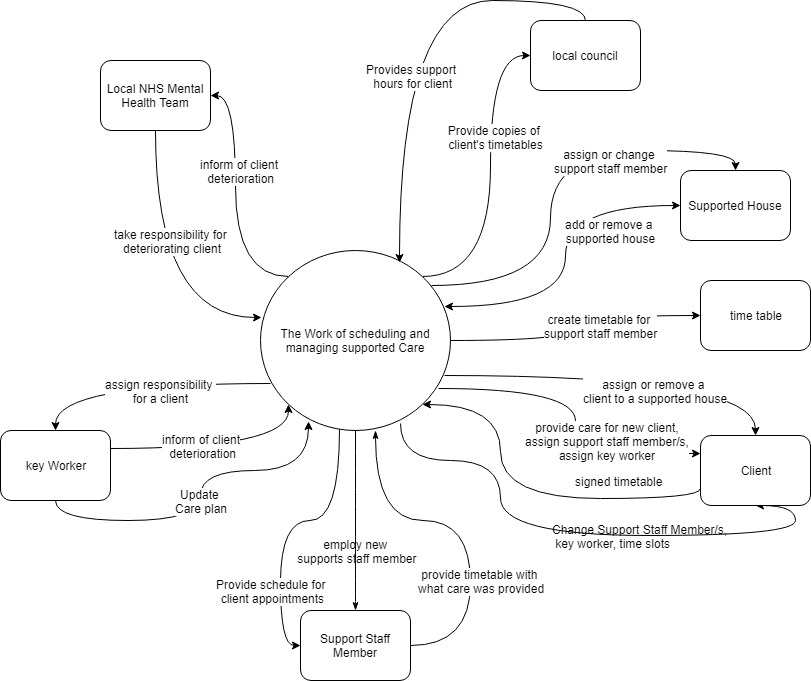
Using the Volere template would improve the likelihood of creating a solid base for the project and if done correctly, ensure that the software produced would meet the requirements and expectations of the charity. If the Volere template were done, in-correctly much time would be spent creating a software solution just not to the actual problem. So, around a third of the project's timetable was devoted just to the creation of this document.

The first phase of this (Robertson et al., 2013) defined as the ‘blast-off’ phase and related to defining the scope, constraints, and stakeholders of a system.



(Fig, 01 - stakeholder’s model).

Figure one shows a breakdown of the system stakeholders and users. This information came from personal experience with the charity and informal talks with other staff members. The work on the stakeholder’s model happened concurrently with defining the scope of the current system.

****

(Fig 02 – work event model)

Figure two showed the work event model and was used to create an understanding of the inputs and outputs of scheduling and managed supported care. From this list, several flow diagrams around events that were critical to the current system but also relevant to the system that would be replacing them. These were created by talking to the staff members who were currently responsible for them.

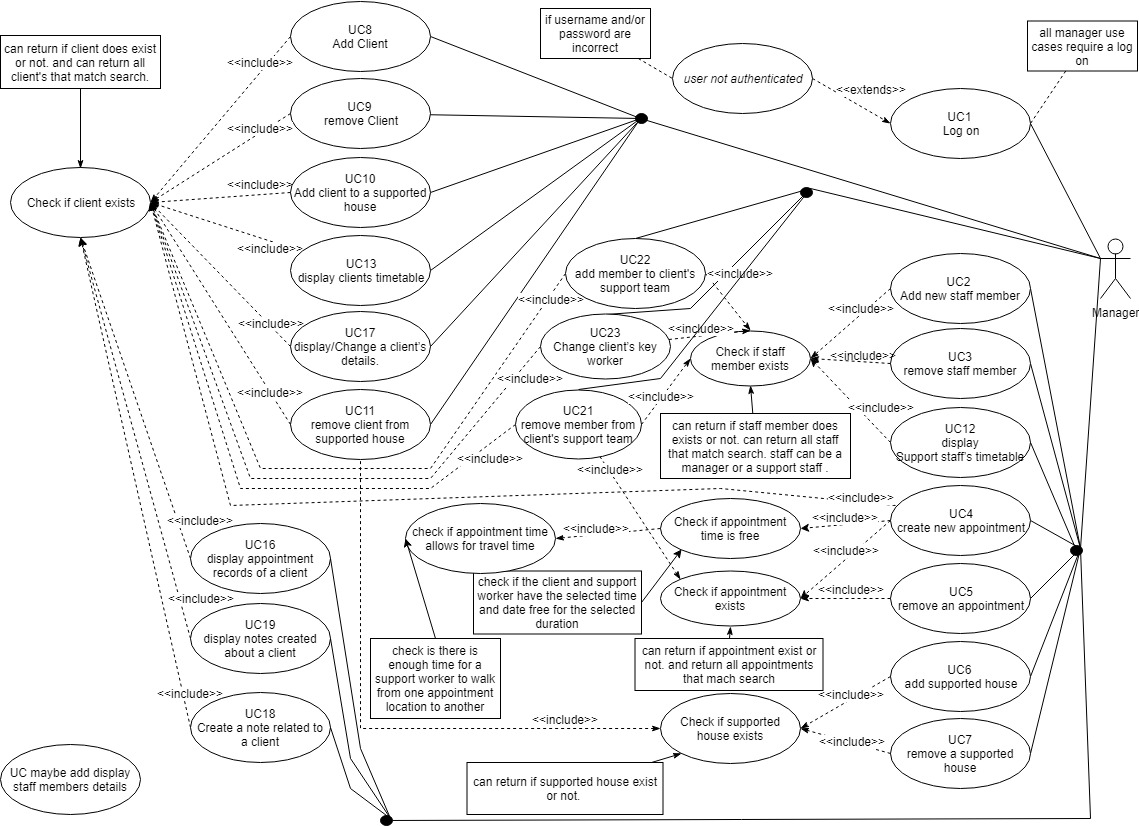
(Fig 03 – A flow diagram around a support staff member meeting a client.)

Figure three shows a flow diagram around a staff member and client meeting using the current paper-based system; this is useful as while the paper-based system is to be replaced, the actions are the same. The rest of the flow diagrams are in Appendix A from page 10.

Using business use cases also helped with getting staff to think of their job as a collection of events. This help focused the more experienced support staff, and managers that have such an understanding of their role they worked without having to think through each step. The Business Use Cases helped them think more about small events and decisions.

This work also altered the perception that the support staff worker was the primary user and benefactor of the system. It would be the managers that would act as the crux of the system. If the managers cannot effectively use the system, it will not function correctly. With this realisation, more time was allocated to better understanding the work that managers must accomplish for the system to succeed.

With this preliminary work completed, the design of the solution could start. Beginning with the product use cases for the manager, support staff member, and client. The approach used was to create a use case, writing up the detailed use case scenario and come up with more use cases based on the business events, and discussions with the charity’s staff members.



(Fig 04 – Manager product use case.)

Product use cases were particularly useful for discussing system functionality. An example of this was, one of the charity managers noticing that while the system had use cases around adding and removing Support staff and clients. It did not have any around adding and removing managers and the restrictions this would need. For example, could a manager remove their account? In this case, the answer was that no manager should be able to alter their account.

However, as figure four shows as the complexity of the system increased the visual product use cases started to become unwieldy and ended up requiring much more explanation of what it meant. Because of this, more detailed use cases were mainly used for the development process, and less specific use cases were created as needed for discussions with charity staff members.

The format of the Individual product use cases is a slightly customised use case based on the Volere template but some elements like alternative if flows are from the book Larman, C. (2005) Applying UML and patterns.

An example of this is on page thirty-three of appendix A. UC23 Change client’s key worker uses a letter with a decimal increment in the main success Scenario to denote a branch in the flow. In this case

5, The system asks if the new key worker is part of the current support team.

If the manager says no, the system displays all eligible support staff members (support staff with the same area of experience. general mental health, learning disabilities or forensic mental health).

A5.1 If the manager says yes. The system displays the current support team of the client.

This alteration allows the use case to handle simple branching in the Main Success Scenario where the impact of the branch did not need an extension.

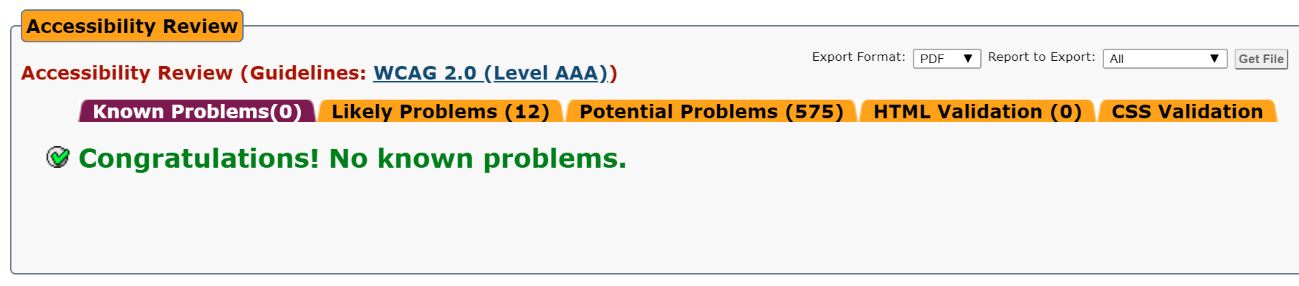
With the detailed use cases finished, they could be used to elicit the functional software requirements. After completion of the SFR, next was the Non-functional Requirements.

The first set of non-functional requirements relate to the Look and feel of the web application. Making sure the web application was in line with the charity’s website and had a professional style with functionality over aesthetics. The plan agreed with the charity was to create several sketches of the web application and from their refinement create a wireframe and then implement the final approved option from the refined wireframe.

Another part of this was the need for the website to make use of responsive design. The charity was planning to implement a bring your own phone (BYOP) policy. So, the site would need to conform to a few different screen sizes.

The use of BYOP is a choice of management in the hope to not incur the costs of outfitting the support staff with phones if this system is to be implemented. Through talks, it was agreed that the website would not need to support phones with operating systems over five years old, e.g., older model phones that could not update to the newest software versions available.

A vital usability requirement is that the web app meets the Web Content Accessibility Guidelines with a minimum of WCAG 2.0 (level AA), which is now a legal requirement under the Equality Act 2010. An online WCAG checked would be used to check if this requirement was met.



(Fig 05 – screen capture of the final website passing the WCAG 2.0 (level AAA))

At this point, the requirements document was complete and reviewed with a few of the charity staff members from support staff to the managers. The response was positive with the belief that if the final system mostly mirrors the requirements document, it would satisfy the charity's needs. Another benefit of this process is that it involved the charity staff members in the development of the system, giving them a sense of ownership and that the design of the system was a collaborative event this engagement meant that designing the system with staff became much more straightforward.

### Part 2 - underlying object representations and appointments.

With the completion of the Volere requirements document work could start on the first iteration this began by analysing the requirements document and creating a Domain Model along with its Glossary.

Using the domain model, it was possible to create the first analysis class diagram along with its corresponding methods and glossary. From the analysis model, use cases, and SFRs, the system operations were designed.

An example of this is on page two of appendix B. UC3 remove staff member; The system operations are shown with their pre and post configuration and relevant models to explain messages between the relevant object and the states these objects change two. Each system operation follows this pattern.

At this point, work started on the implementation of this system using IntelliJ as the IDE and Java. The work started with the creation of the abstract Person and abstract Staff class that inherits from the Person Class.

The next class created that implemented the abstract Staff class was the SupportStaffMember class. For each concrete class, a JUnit test suite was created. These test suites allowed the system to tested whenever a change was made, and so be sure that the system functioned as before the changes were introduced. Moreover, as each test suit related to a specific class and its methods tracking down bugs introduced with system changes was simplified; this is referred to as regression testing and is the central part of minimising system bugs during the different incrementations.

The tests themselves were written on the boundaries of inputs or when an objects state changed. Other tests were created to check common problems like the implementation of equals.

The implementations followed the class and Sequence models until these started producing problems. The original plan was to use actors as classes as is taught in TM354 as one way to implement use cases. The problem was as new classes where created, the overall complexity of messages passing between objects was continually increasing.

An example of this was with the implementation of charity class to hold references to Staff, Client, and SupportedHouse objects. When the Manager class initiated a message to add or alter an object, it needed to consistently first message the Charity class and send references to that class in most of its messages.

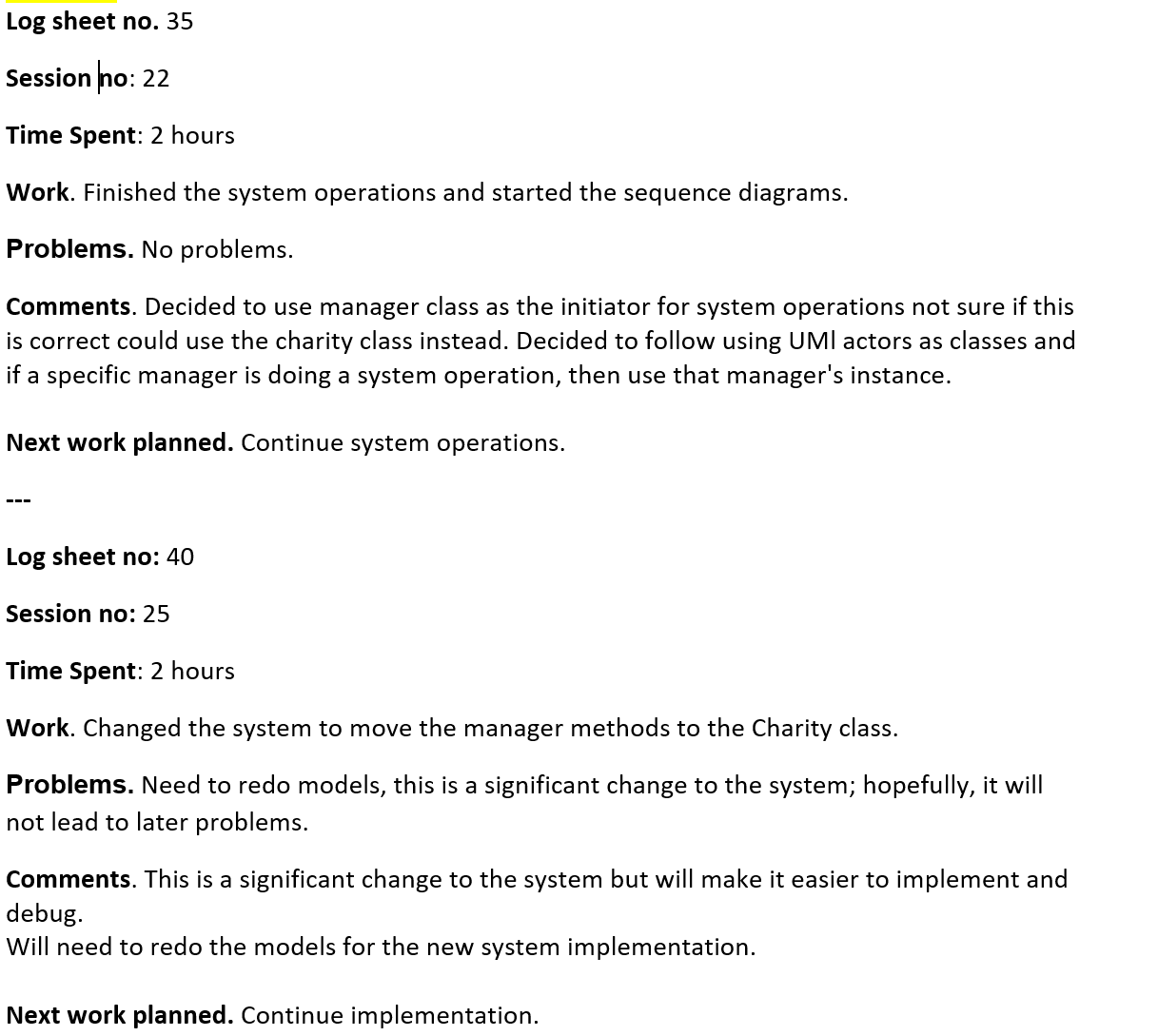
One option was to have had the Manager class handle all implementation of a sequence diagram, but this would violate the Law of Demeter by giving the Manager class to much knowledge of other classes.

In the end, the design was altered to use a central class implementation using Charity as that central class. The drawback of this is now the Charity class has become overloaded with methods, but this has made the iteration easier to implement.

This mistake came from seeing the manager person, actor, and class merge into one. A manager at the charity controls the appointments; the actor representing the manager initiates the Appointments, so followed into a manager class creates the appointment objects.

Some of the other design decisions made were to use an interface for appointment control between the Client and SupportStaffMember.

The Appointment system was implemented through the Staff Class, so if need be, the system can be changed to accept Managers being assigned clients and appointments. This decision was a point of discussion with the charity. As managers are not meant to take appointments, but if there are no free Support staff, they sometimes do. Below are two extracts from the project journal around these changes.



I also replaced most of the class constructors. With factory methods. This was recommended in the effective java book. As an example, this allowed me to control which of the abstract class staff’s subclasses were created based on the parameters passed during construction.

Another design choice was to use design by contract for client-side validation. For example, the front-end interface will validate the date of birth of a Support Staff member before that data is sent to the backend using JavaScript so, the server will not need to check that the staff member is over 18 years old. This would contrast with defensive programming where the pre-condition would still be checked by the server.

This work concluded the first iteration. Overall the work in this iteration went well but the design mistakes introduced during this iteration caused several problems later.

### Part 3 - website design and restful links to Java backend

With the completion of the first iteration, work began on the second iteration this focused on the website construction and connecting it to the Java backend. The first part of this was performing an in-depth look at Spring Framework; this also revealed a substantial mistake in the previous project work.

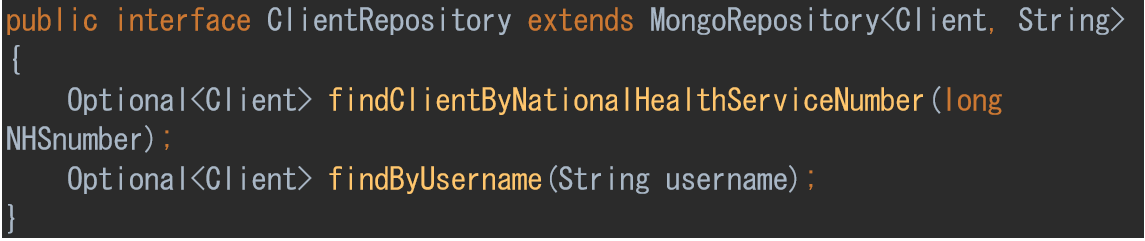
While researching Spring framework through the book Spring in Action proved its benefits, things like dependency injection and automating server applets along with database implementation.

The error was underestimating the number of changes, that would be needed to implement Spring into the current project code. This error meant that most of the class designs from the first iteration would need to undergo substantial changes in this one. Another part of this problem was while Spring in Action provided examples of the benefits of the Framework, it was not so obvious how this could be implemented. I continually thought that I could read my way to a solution. Until it became apparent, I would need to make substantial changes to use the Spring Framework.

However, before work could begin on implementing Spring and making the necessary changes, Spring Framework would need to know which database was to be implemented. Looking through the Spring database implementation, it would be possible to use a non-SQL database and have Spring handle all the database’s implementation as well as saving and retrieving data. Because of this, MongoDB was chosen with the hope it could save time by not requiring the design of an SQL relational database or writing SQL commands.

To implement this in the project, the Spring tag @Document would be added to the class to be stored. The chosen classes are Staff, supportedHouse, and Client. Next, two new classes for each class that would act as a database document needed to be created.

Firstly, to access the saved documents, each class that implemented the @Document would need to apply a customised version of the Repository interface.



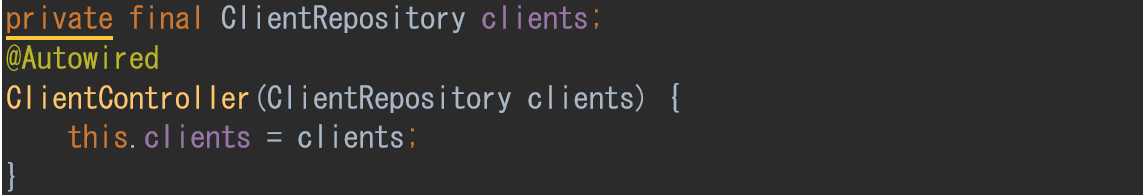
This interface is used to configure what parameters can be used to search the database. For example, findClientByNationalHealthServiceNumber is automatically implemented based on the method name; this could be changed to findClientByFirstName to search the database for clients that match the name. Both methods make use of Optional<Client> which makes it easier to know if a Client object was located without null testing.

The second new classes are ClientController, StaffController, and SupportedHouseController. Each class used the tags.

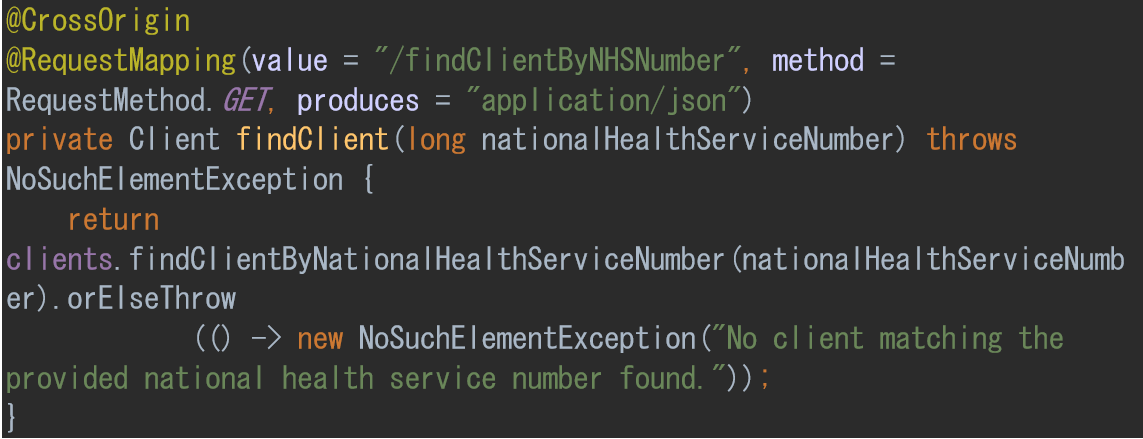


The RequestMapping tag is used to tell spring that this class will respond to URL commands on the “/” path. Moreover, the @RestController allows Spring to serialise return objects into HttpResponse.

Another essential part of this is the dependency injection using the @Autowired tag.



This tag tells Spring that the interface ClientRepository will be accessible to this class without needing to be passed a concrete referenced to it. Acting almost like a static object but being initialised by Spring.

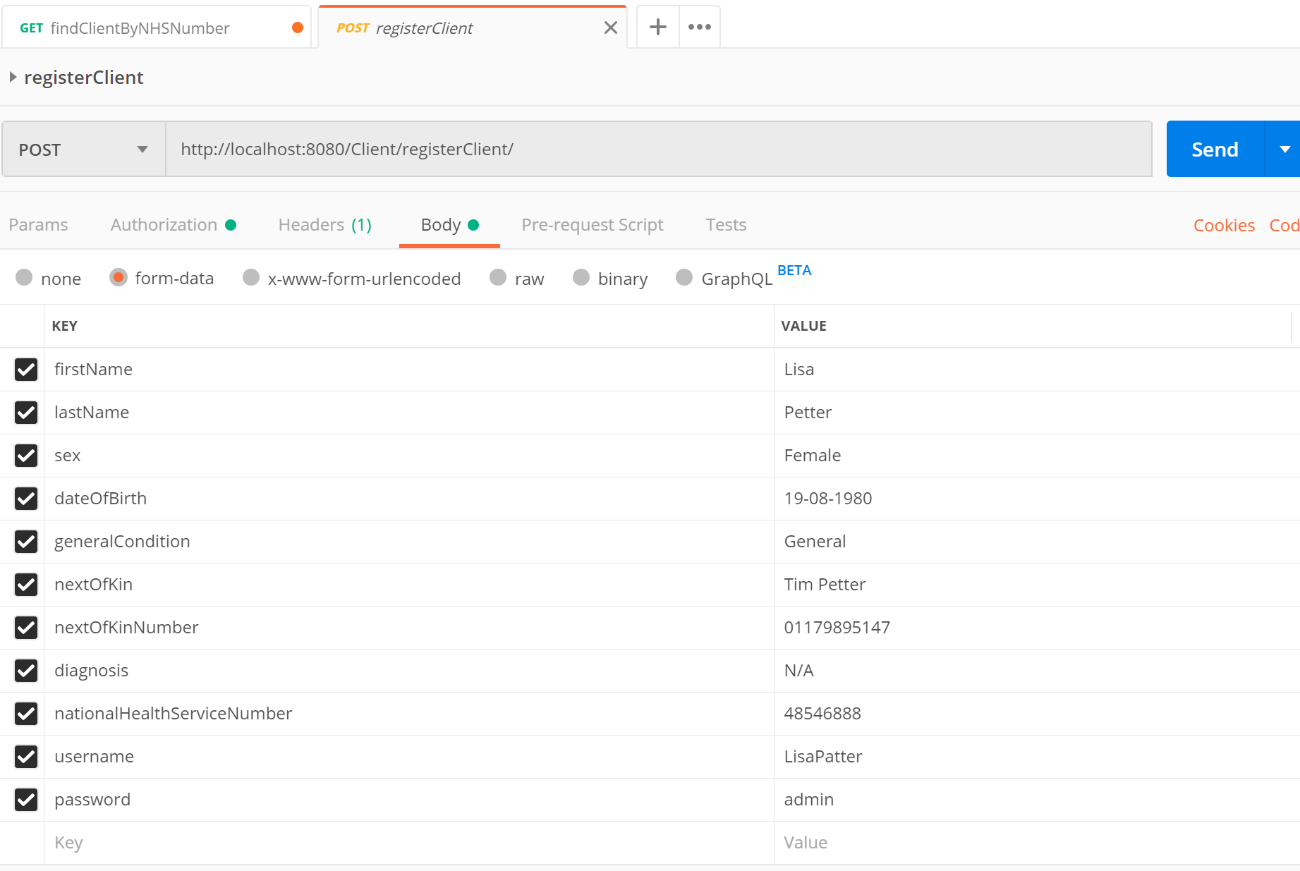


Each method in these classes can be made to respond to HTTP Methods using @RequestMapping. The value = "/findClientByNHSNumber" is the path to respond to. And the method = RequestMethod.GET tells Spring that this method should only respond to GET commands.

This method finds a Client object and returns a JSON object to the frontend or returns the thrown error. The clients variable made using the @Autowired tag is used to access the database to search for a Client object with the matching National Health Service Number.

At this point, some of the HTTP Methods could be tested using the program Postman.

Postman allows the URL’s to be tested directly.



(Fig 01 – postman setup to test the register client URL)

Figure one shows the postman setup to test the register client URL. This program will then display the HTML code of the request, 200 for completed or a number that can be checked if an error is returned for example 401 would be an authorisation error. These tests can be used to check how the system responds to incomplete data being sent or bad data like the date of birth parameter being sent with the string fish.

These tests revealed a problem with how Spring converted Java objects to JSON objects. Specifically, when an object held references to other objects. That also contained references to that object. For example, in the class Client is a collection of Appointment objects, which themselves hold a reference to the Client object. In Java, this was not a problem, but when Spring converted this into a JSON object, it would cause a stackoverflow error.

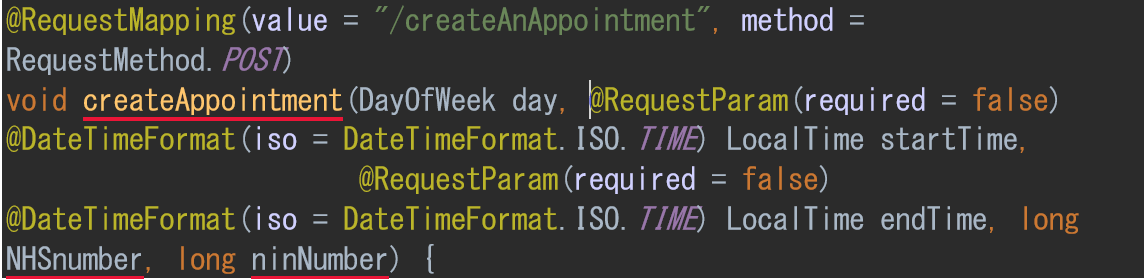
This error was challenging to track down and fix, but through reading the Spring documentation and some stackoverflow forum posts, this problem was called a circular dependency error and could be fixed by using a mix of @DBRef(lazy = true) and @JsonIgnore. This problem highlighted my inexperience with databases and specifically MongoDB. MongoDB was chosen to save time on database design, but in the end, it was a costly choice with the amount of time lost fixing errors that may not have been necessary had the time been taken to create a SQL relational database.

The most substantial changes to the project came from updating the Charity Class to use Spring. Firstly, I renamed the class to CharityController as that would be its new role.



this class now has access to all the Repositories and is used for methods that will need to access two or more database documents or perform some configuration of Spring.

An Example of this is the createAppointment method. this method uses both the Client and Staff Repository’s. and uses some new tags.



The @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) allows me to tell Spring what format of the date the method will allow as a parameter. This method uses the RequestMethod.POST, so Spring takes a JSON object in the required format and converts it into its Java equivalent. There are some problems with these conversions; for example, DayOfWeek in java has an enum of MONDAY, TUESDAY, et al. but Spring converts this into a number when sent to the front-end. Small oddities like this create problems that can be difficult to track down.

The body of the method checks that the passed ID’s match a client and staff member if either does not match an object then an error is thrown.

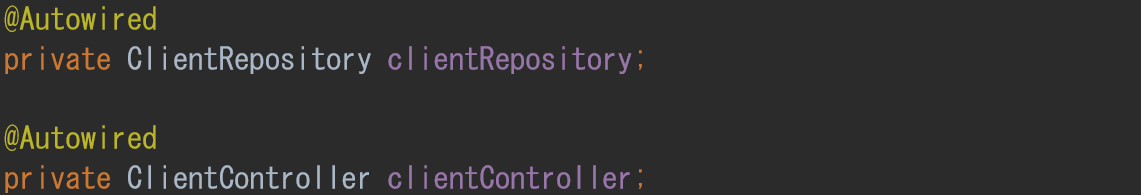


Next, the client and staff objects check if they are free on the specified date and time by using their checkFree methods. If either is not available an error will be thrown. If the slot is free, then an Appointment Object is created, and both the Client and SupportStaffMember objects record this new appointment object. At this point, both repositories are updated with the new client and staff objects.

Another problem implementing Spring was it made most of the written Junit Tests that needed to interact with Spring controller classes or variables stop working this was because Spring had a specific setup for JUnit tests. For a test that needs to interact with Spring, the following tags are needed before the Class.



Then dependency injection is used for the required Repositories.

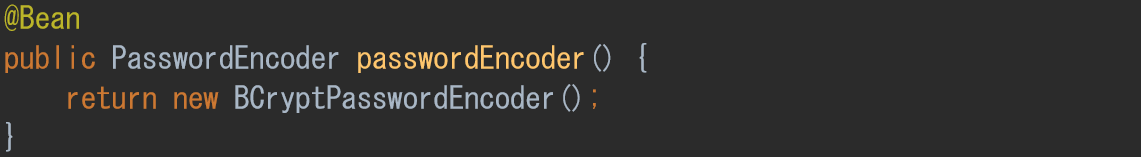


The tests themselves are very similar to without Spring.

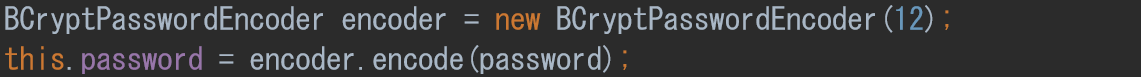


Using the @Test tag and the @Autowired classes along with standard asserts. These changes were not too complicated to implement, but this did require much time.

With this complete and the database and HTTP methods, now working. The next step in this iteration was to begin implementing security features for the project. Through reading the Spring documentation, it would be necessary to implement the class WebSecurityConfig and then configure it to meet the specific needs of the project. The first part of this was securely storing the user password in the database. Using Spring, this was accomplished by creating a bean method called passwordEncoder.



This method will create an encrypted password based on BCrypt. That is implemented in the classes that will store the password. In this project, that is in the Person abstract class constructor.



Using BCrypt with HTTPS and TLS should provide an acceptable level of security for the web application to take and store passwords safely.

The next security implementation was around setting user roles and using that to limit backend access and allowing the frontend to be configured based on who was logged in. By reading the Spring documentation, showed that the configured method of the WebSecurityConfig class could be used to restrict access points to user roles. Using the HttpSecurity object as follows.



using the. antMatchers perimeter access can be restricted on a selected path to a specific user role. .antMatchers("/Client/findClientByNHSNumber/").hasRole("MANAGER")

In this example to send requests to the /Client/findClientByNHSNumber/ path the logged-in user must be a manager. The path /Staff/\*\* means that any path within the Staff Controller requires a manager role.

The next step was to link a role with a user based on which class that user was assigned Client, Staff, or Manager This required a new Class called MongoUserDetailsService that has access to all the repository’s and is used by Spring security. This class has one method loadUserByUsername.

This class first checks if the person trying to log in has a legitimate username.



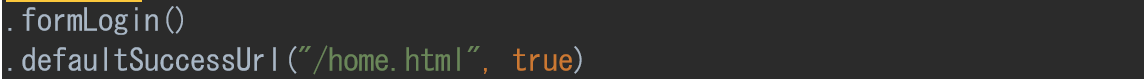
If they do then depending on which repository the username was from the method will assign them an authority role of Manager, Staff or Client. in a new object called RequestUserDetails, this holds the Authority, username, password, first name, last name, and their ID. This object RequestUserDetails is a Spring class that has been further customised to hold the first name, last name, and the ID.



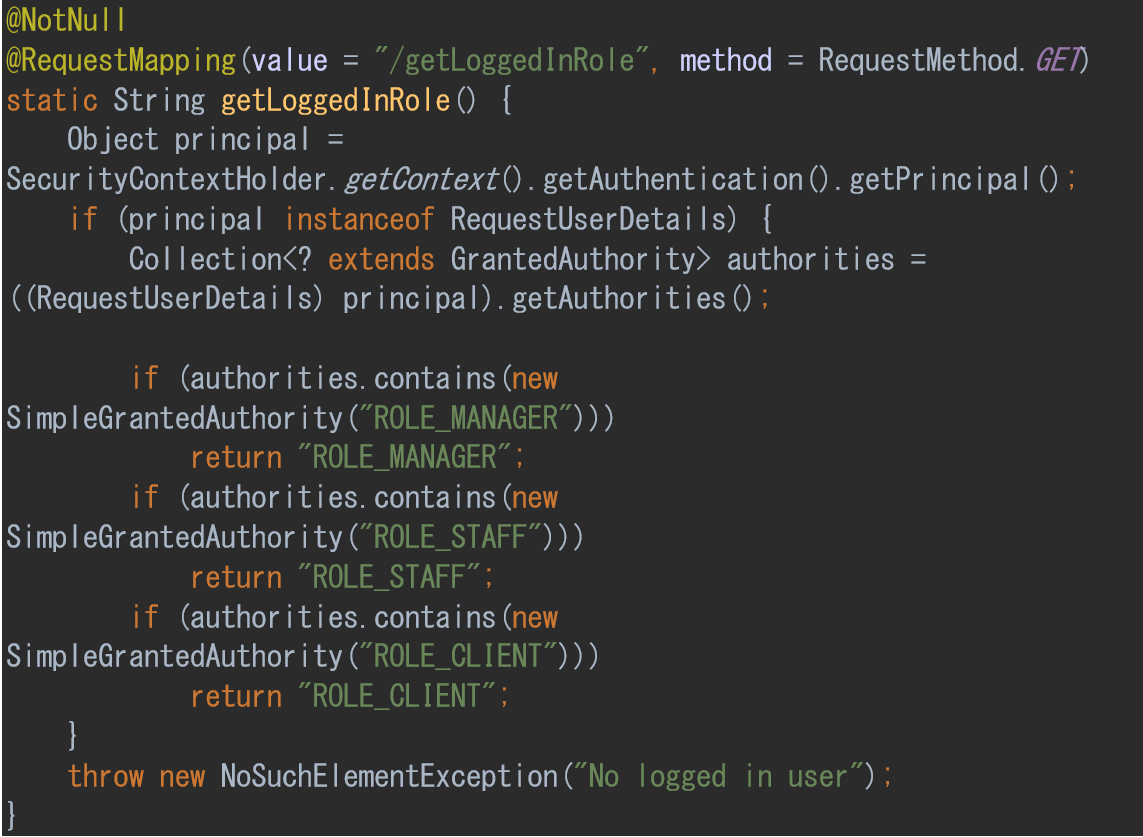
if the username was not found then an error is thrown.



With this complete, Spring provides a Login page for authorisation that can be further customised later if needed.



Now a user must log in to the website and based on the user’s authority; the available functions will be specific to that role. When a user has logged in their role is stored and can be retrieved using the getLoggedInRole method from the CharityController, this method is also static so that other methods can check the logged-in users’ role.



At this point, the URL links have been created and tested. The system has different user authorities and meets the requirements of most the use cases but without a user interface.

With the backend in a good state, work could begin on, implementing the frontend and linking them with JavaScript as the website design work was not complete at this point. Only the functional aspects of the website could be started.

One design choice that was already agreed on was to use a calendar for the appointments. Through research, a suitable open source calendar plugin called FullCalendar was found. To use FullCalendar all that is required was to create a div with the id calendar and link the CSS and JavaScript files in the header. That could then be used with JavaScript to render the calendar. JavaScript’s document ready function will render and populate the calendar with appointment and appointment meetings data.



This function makes use of the /Charity/getLoggedInRole/ path to get the role of the logged-in user. That data is then held in a variable called userRole and passed as a parameter of loadCalendar.

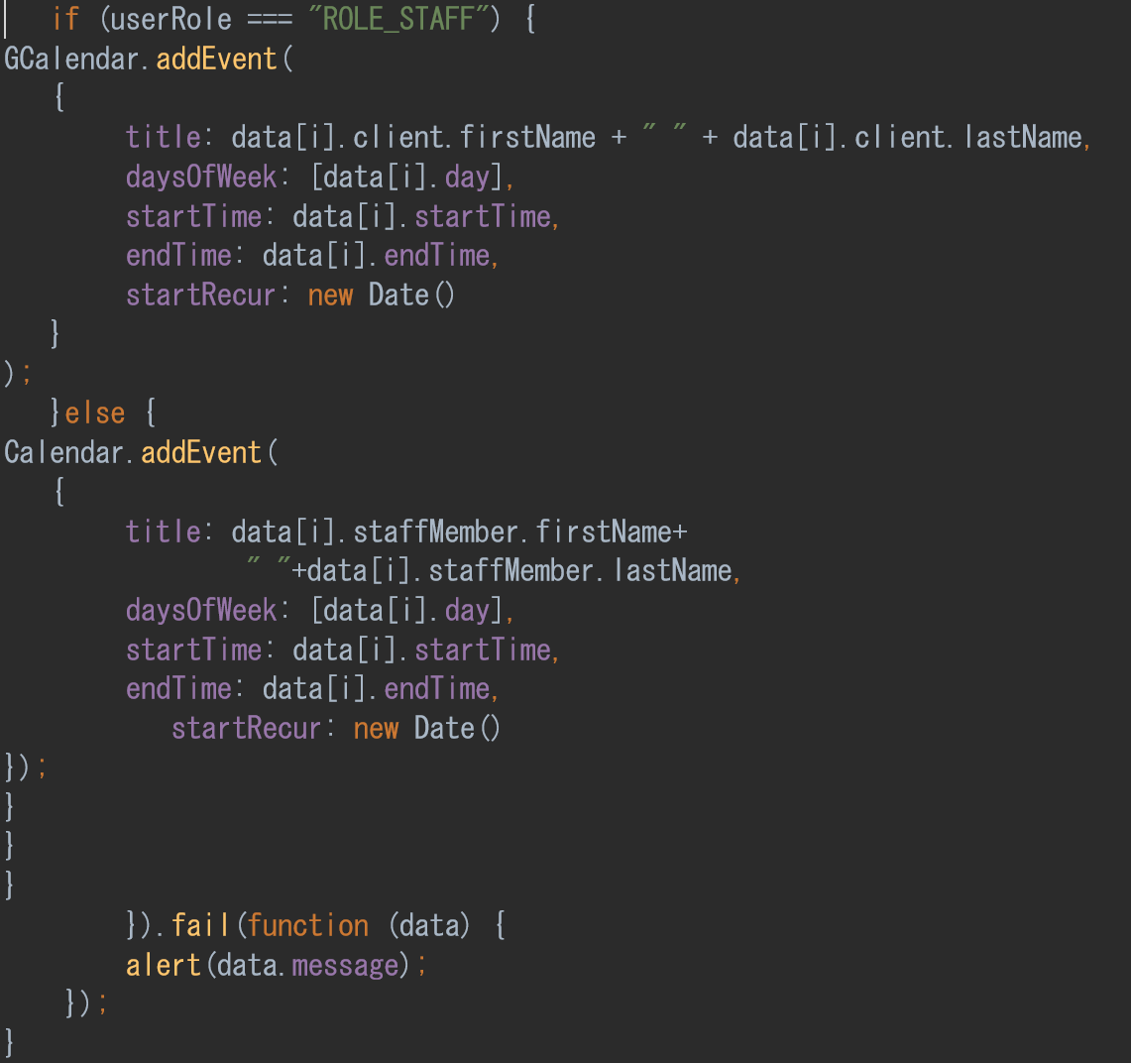
The function loadCalendar configures the calendar then calls the updateCalendar function. If the logged-in user is not a manager.



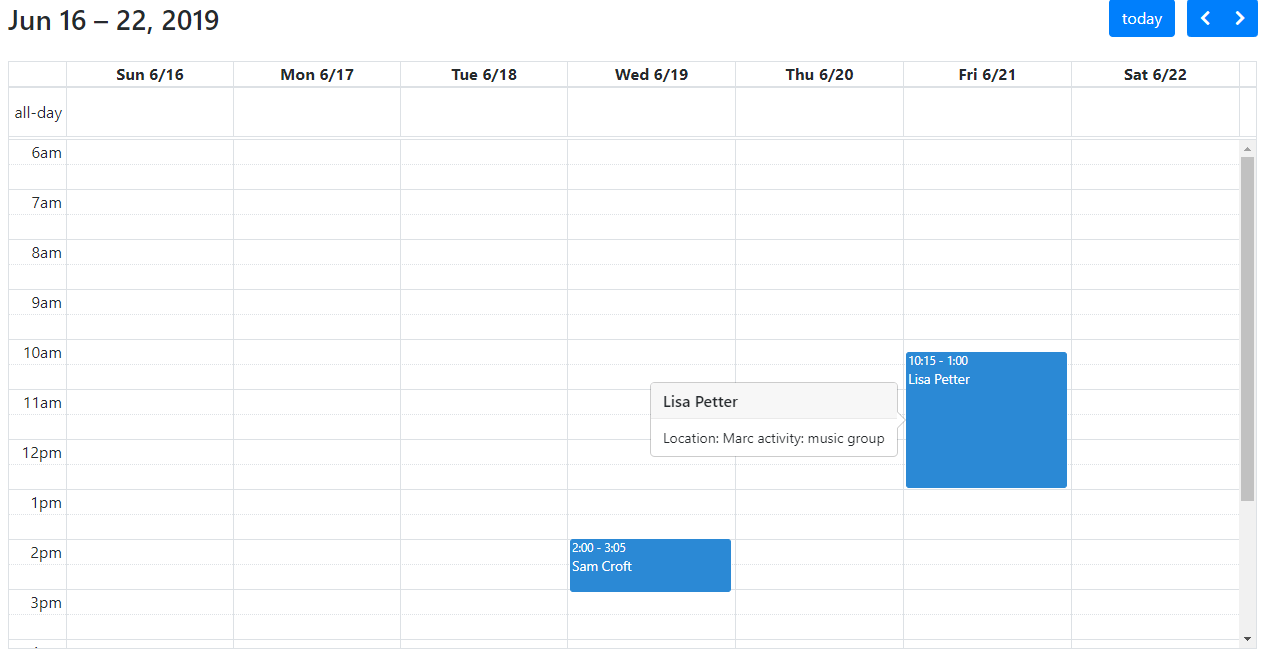
The function updateCalendar allows appointment data to be added into the FullCalendar

this is done using the /Charity/getAppointments path that returns a JSON object of the appointments that specific logged-in user has. The function then iterates over the JSON object and using the addEvent function creates an event on the calendar.





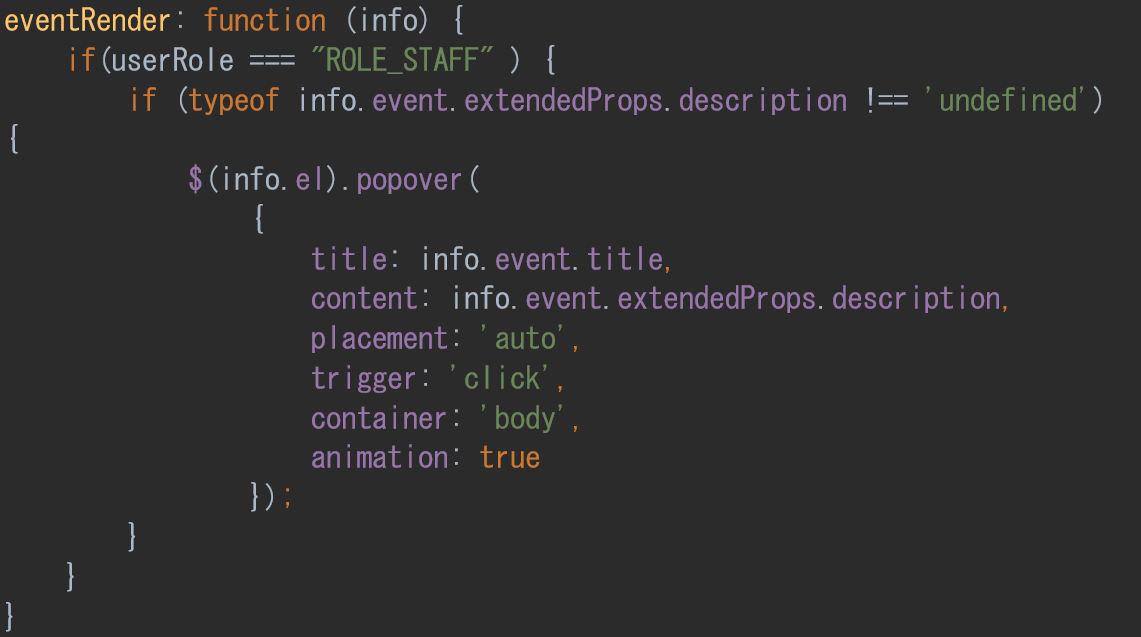
This function is set up to work for a Client or Support Staff Member. Additional functionality for Support Staff Members is that Appointment Meetings are also added to the calendar. Appointment Meetings represent appointments that have happened and been recorded between the Client and Support Staff Member. So, on the calendar appointments shown after or on the current date are the Appointments the user has coming, and appointments before the current date are Appointments that have a recorded meeting.



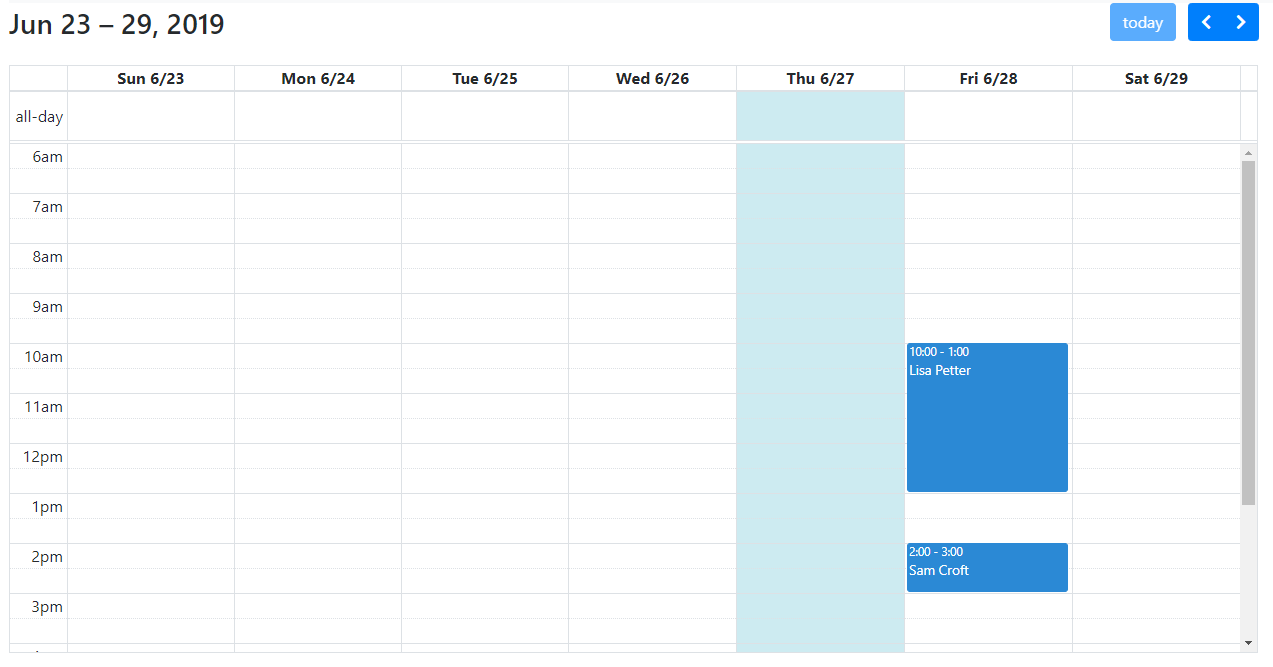
(fig 02 - the calendar set to a past week.)

Figure two shows a previous week that has two recorded appointment meetings. Both are displayed with the time the appointment meeting started, finished the who the client was. Another function is if the user clicks within the blue box, then the notes and location of that meeting are displayed in a popover.

This is done using the function eventRender within loadCalendar(userRole).

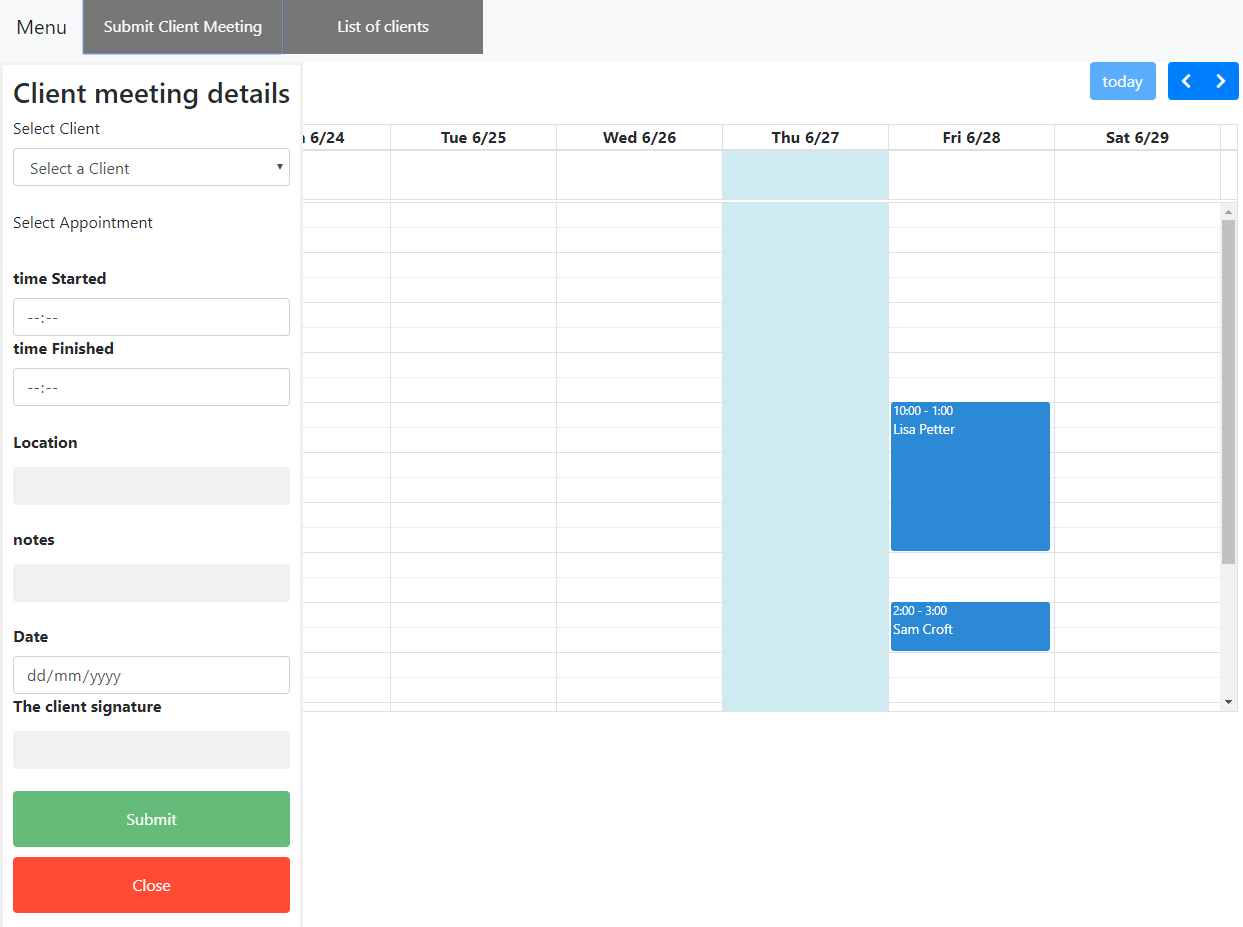


This function makes use of the popover function from bootstrap. If the logged-in user has the role of STAFF, the appointment meeting notes and location are displayed in a popup.



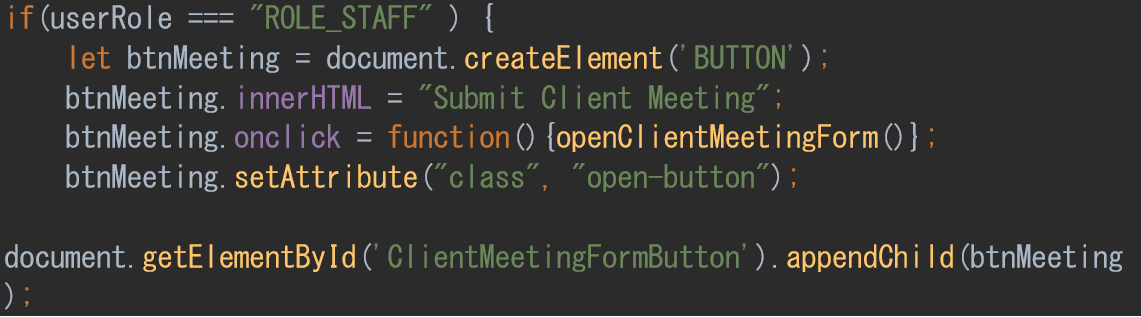
(fig 03 – the calendar is on the current week.)

Fig three shows the upcoming appointments for that Friday, and this also shows that for this week that logged-in user has not attended any appointments. From this, work began to create Staff functionality to record appointment meetings.



(fig 04 – the menu for adding an AppointmentMeeting)

The form inputs are standard HTML inputs with each assigned an ID. The submit Client Meeting button that opens the form is created by JavaScript in the loadCalendar(userRole).

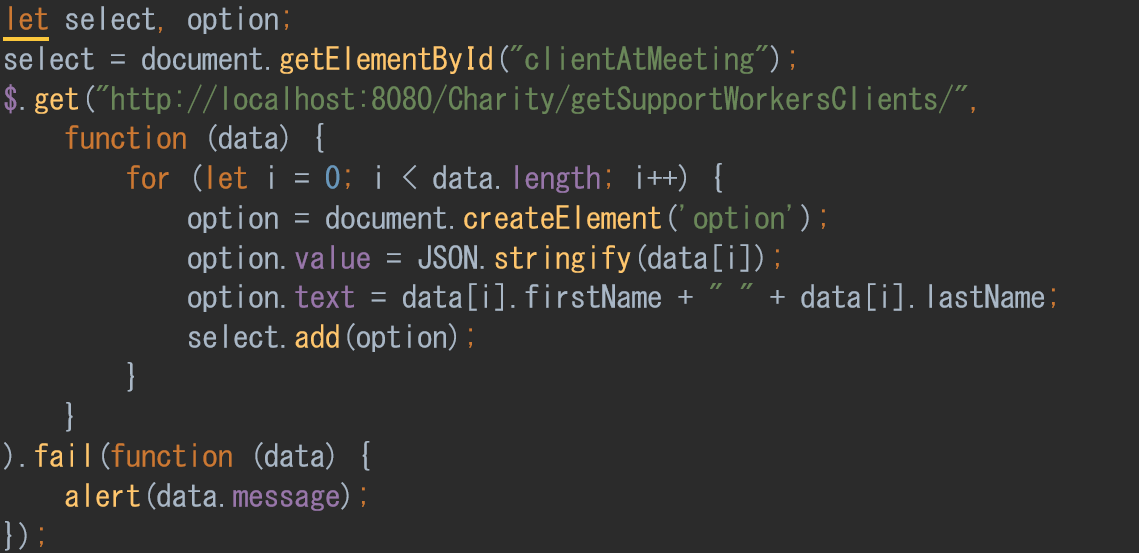


this is done to only show this button to logged-in staff members.

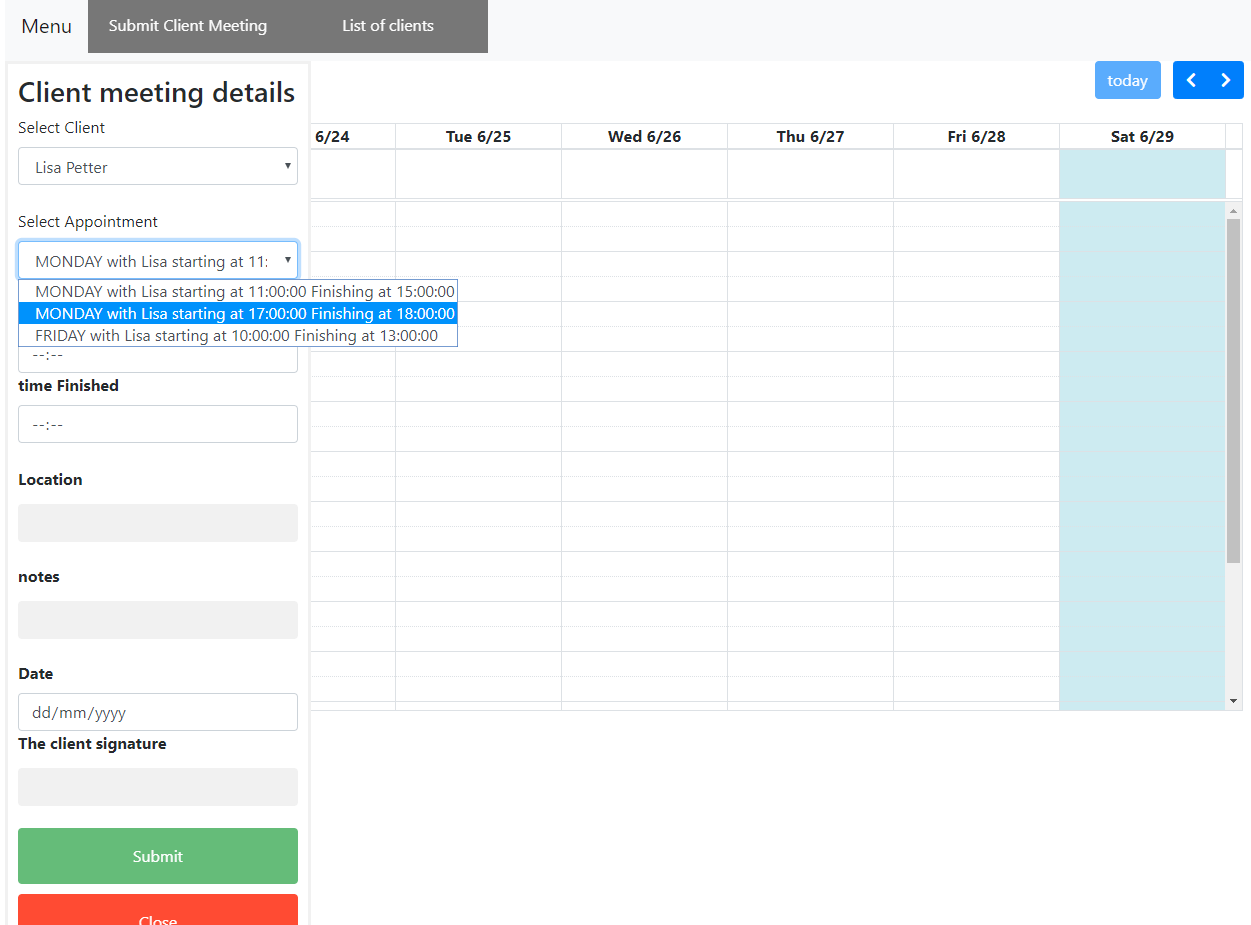
The form itself is set to call a JavaScript function on submit.



This first section of the form allows the user to select from a list of clients that they are associated with. This function makes use of the path /Charity/getSupportWorkersClients that returns a JSON object of each client the logged-in support staff member directly supports.



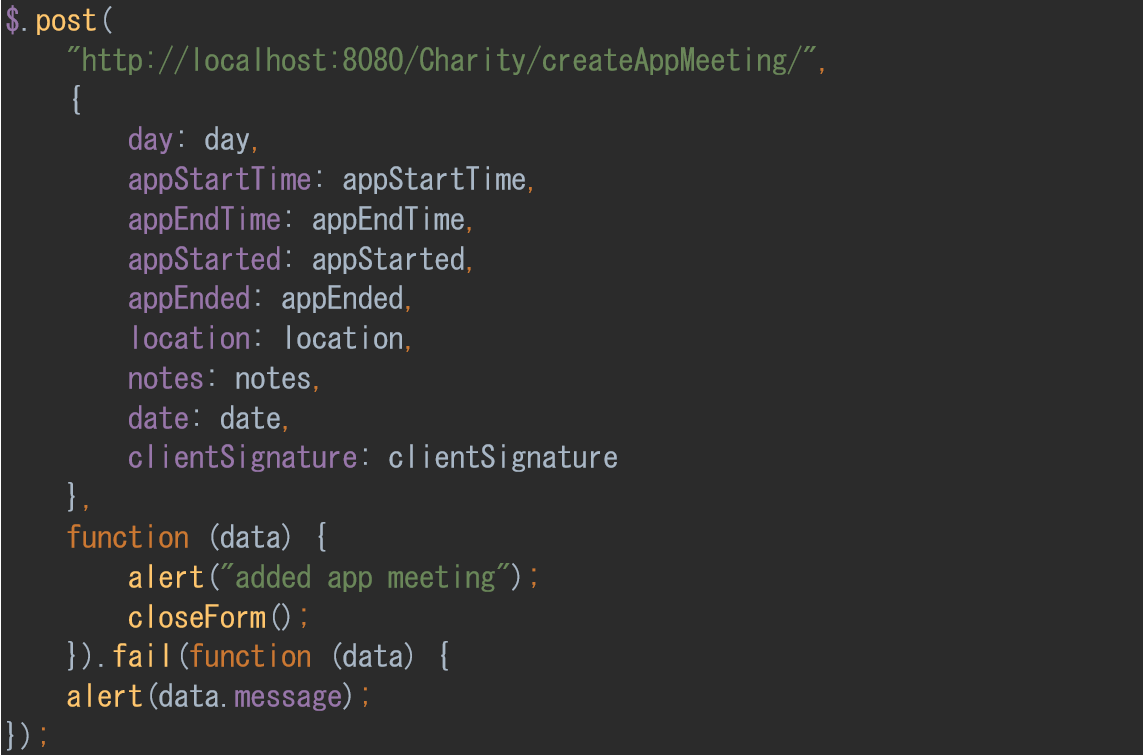
The function iterates over the returned JSON Object and creates the options for the select tag. When the user has selected the client, a new select tag is shown with the lists of appointments that the logged-in support staff member has with that client. This function is very similar to the previous function but uses the path /Charity/getAppointments/ and iterates over a JSON object of Appointment objects.



(Fig 05 – the UI showing selecting an Appointment for a chosen client.)

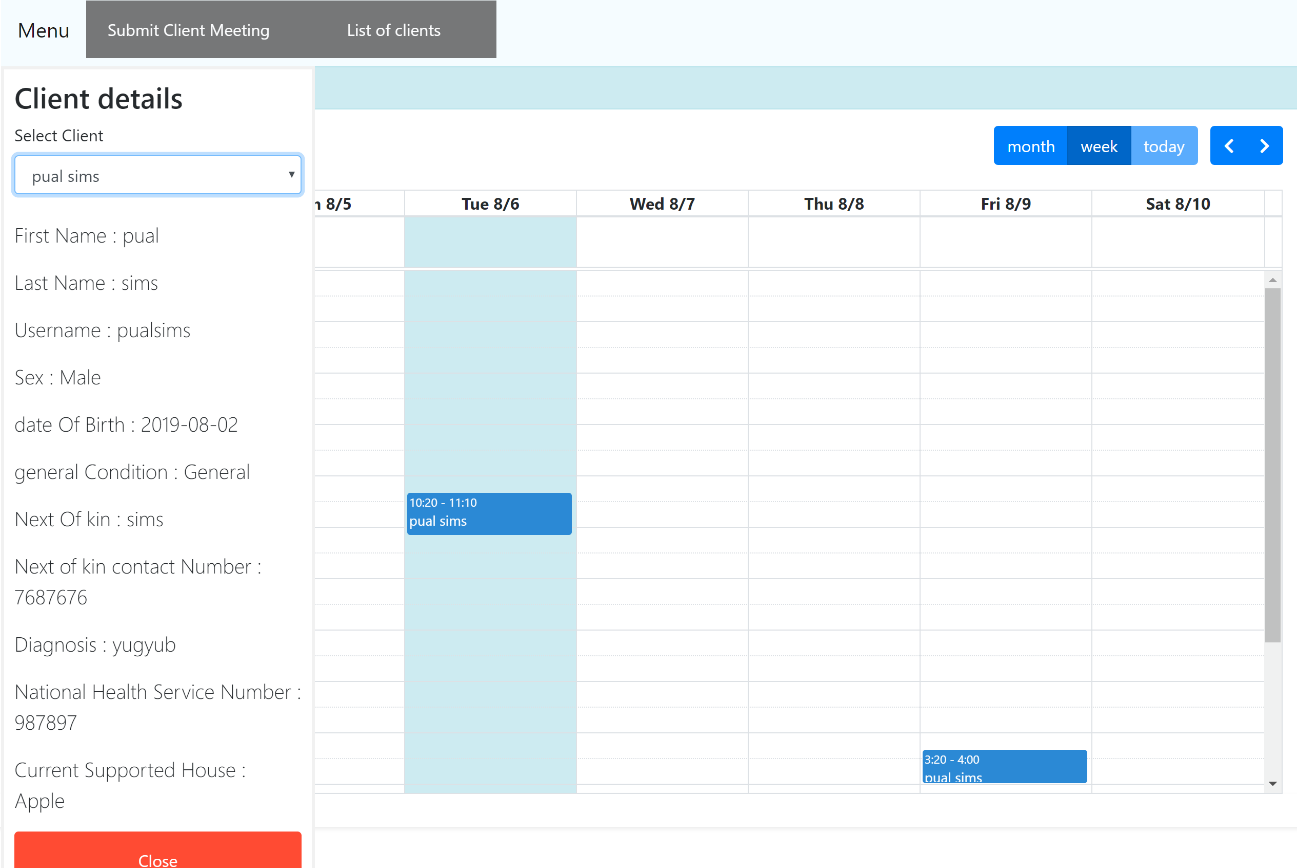
With this done, the user can enter the meeting details and submit them to the server using the submitMeetingForm.

Part of the function submitMeetingForm.



This part of the submitMeetingForm function submits the details of the form to the Charity/createAppMeeting/.

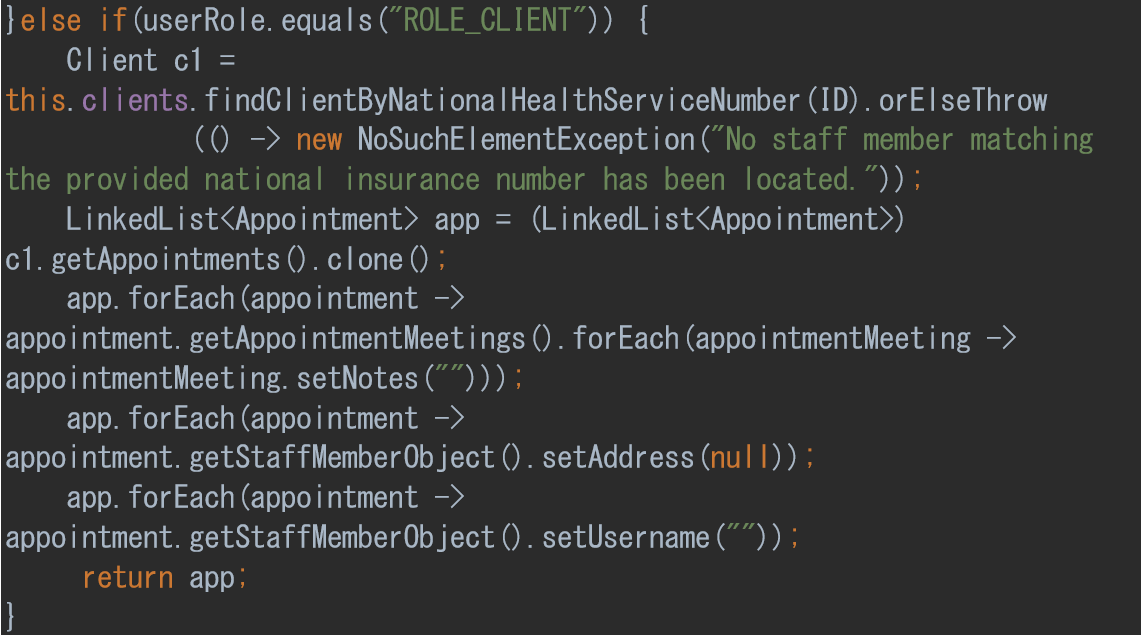
Another completed function is to list all supported clients and allow the logged-in user to access their details this works like the ‘Submit Client Meeting’ in that it gets the JSON object of Clients and uses that data to fill out a form.



(Fig 06 – shows the UI displaying the details of the selected Client.)

Using this system of only allowing logged-in users to select from pre-fill lists of their clients and limiting access to resource URLs by user-role stops the potential for leaks of information.

another way, the system controls data access is scrubbing data. Below is an example of this.

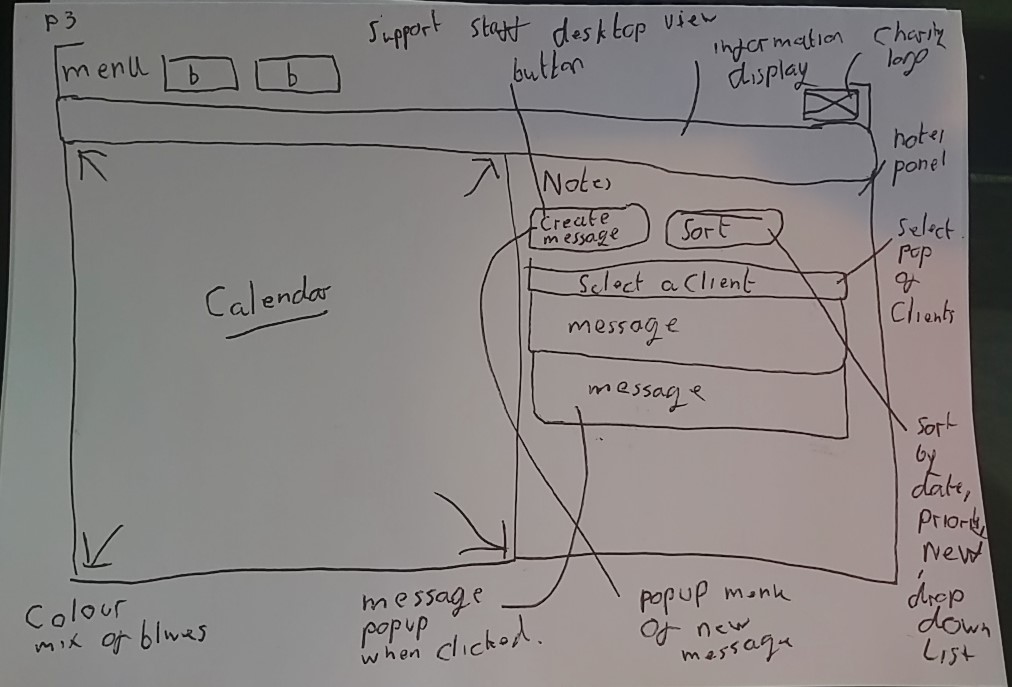


This code checks if the user is a client and if so, will remove staff addresses, usernames, and notes along with using @JsonIgnore to stop returned JSON objects from containing that data. For example, passwords are never returned.

While this does help prevent data leakages, it is a performance drain. To iterate over an extensive array of appointment meetings will always have an O(n). Another option would be to keep another object specifically to return to clients, but that could have update problems and need more memory. The current configuration was considered the best of two not great options. However, with more time, a more optimum solution could have been found.

### Part 4 - website design and note system.

One interesting effect of creating a functional prototype to show the charity what the webpage could do was how the charity staff members had become attached to the functional prototype. This attachment was probably a result of the functional prototype having some design by the necessity of the functionality, and it being interactive.

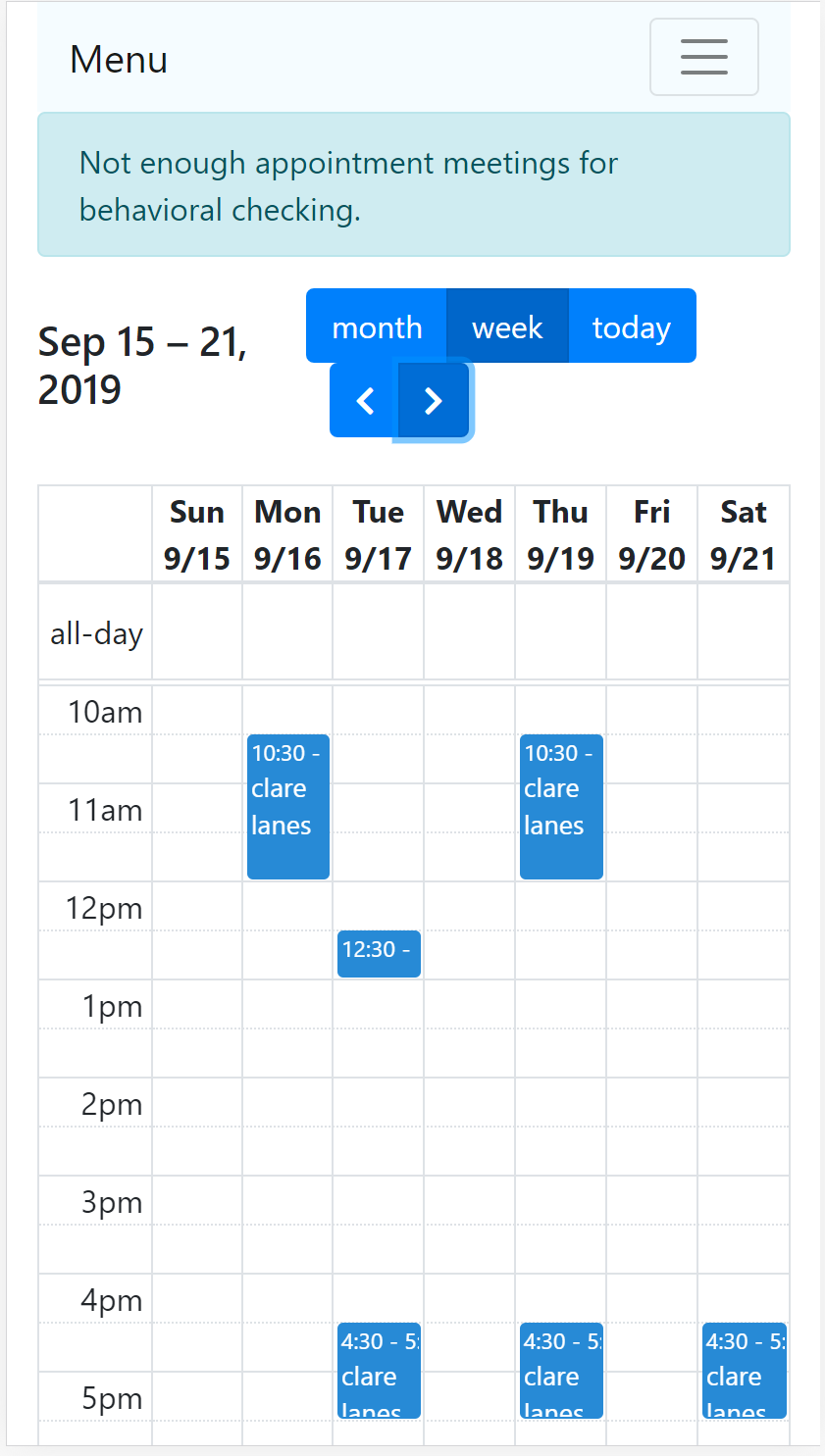
(Fig 01 – a drawling of the support worker’s desktop view UI)

This layout is very similar to the functional prototype in the previous section with just the addition of an information display that will be used to inform the user of something they need to know. Another change is that the calendar will be shrunk to make space for the note functionality.

(Fig 02 – a drawling of the support worker’s mobile view UI )

The mobile view hides the buttons to save screen space and makes use of the screens ability to scroll. With the calendar on top of the screen and the notes below on most phones, this will mean that the user will need to scroll down to see them.

The original plan was to from this make a more professional wireframe. However, time constraints meant that the changes were made to the functional prototype and shown to the staff. This decision to alter the prototype worked well as using chrome dev tools allowed simple changes to be made on the live website. With this complete, the design of the support staff members sections of the website was finalised.



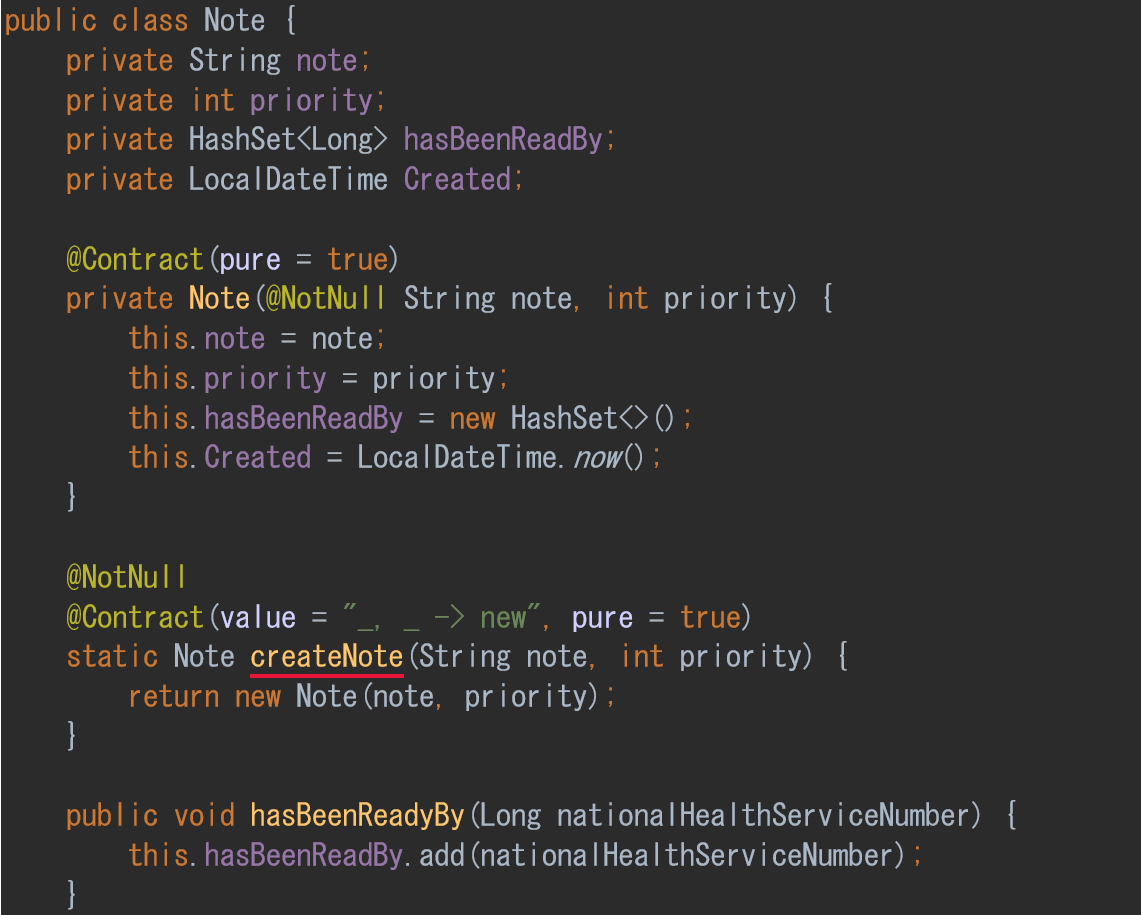
(Fig 03 - screen capture of the website in mobile view)

Figure three shows the finished design for the support staff members website in mobile view.

The two buttons for listing clients and submitting meetings are hidden and can be accessed by clicking the menu button. The messages are below the calendar, not visible in figure three.

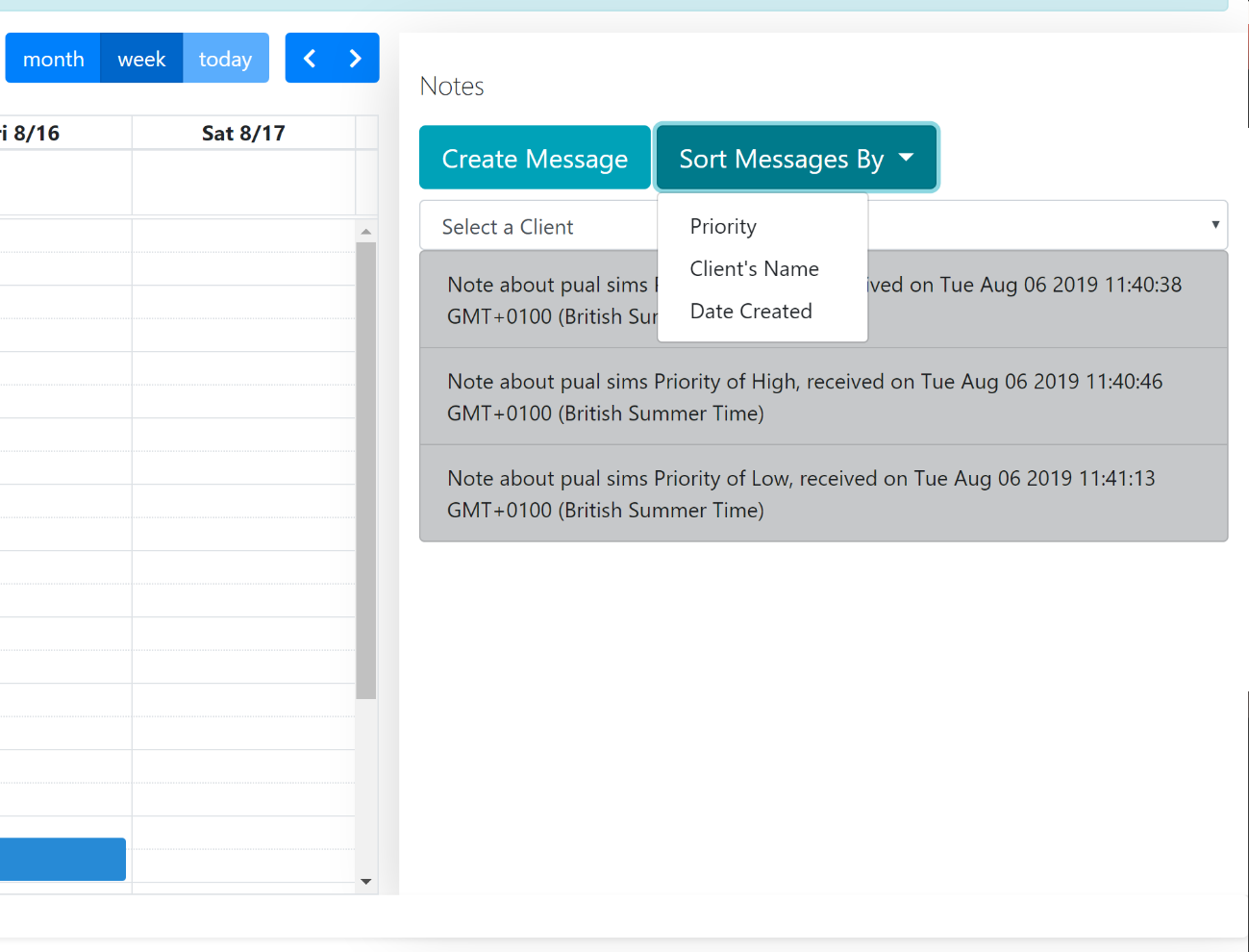
The next sub-system to be developed was the note system which relates to UC18 - Create a note related to a client and UC19 - display notes created about a client. Two of the main parts of this use case is that notes are restricted to support staff assign to that client and newly created notes are pushed to all members of a client’s support team.

the first part of this was to create a note class.



This class is held in the Client class as an ArrayList of Note with two methods createNote() and getNote(). The class itself contains the String note, and priority value, date created and a HashSet of support staff members which have read the note. While the note class is public, the instances are created and maintained within the Client class and stored in a private variable. So only object that can access a specific instance of Client can access its notes. This will simplify permission as if a user has permission to access that Client’s instance they would also have permission to access its messages. Two new methods were created in the ClientController class to link the front-end UI to the new classes and methods.

The layout of the notes section of the website was designed to match the agreed design.



(Fig 04 – the website showing the note section.)



(Fig 05 – the website in mobile view with a message clicked)

In figure four and five, the grey colour on the message indicates that it has already been read by the logged-in user. The Select a Client select box has all the names that the logged-in support staff member supports. Clicking a name will sort the message to only show messages about the selected client.

The JavaScript that enables this functionality is simple but involves much code. First, the function cleans up and sets up the note system.this to functions remove all clients from the Selection box. And then populate it again. then any notes already displayed are removed.With this done all the clients the logged-in user supports are retrieved using the getSupportWorkersClients/ path. This returns a collection that is iterated over. Each iteration checks if that client has any notes usingIf a name has been passed as a parameter from the Select a Client box, then.

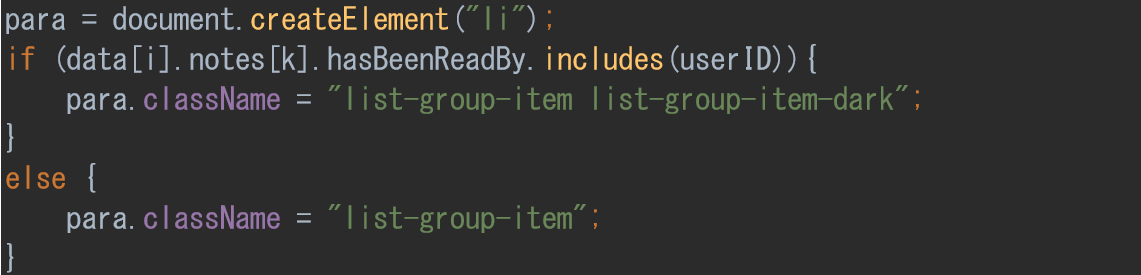


checks this and

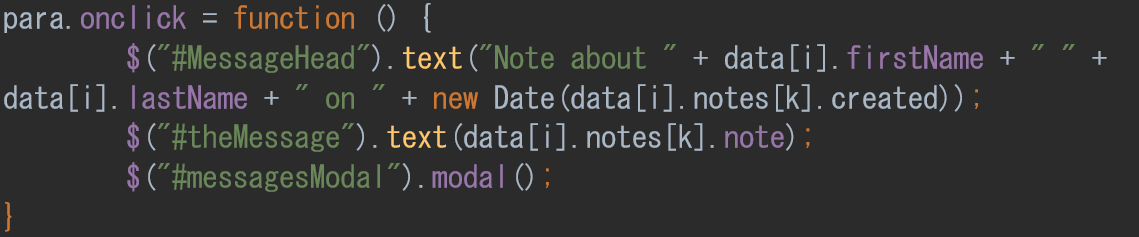


skips any note that has not got that first and last name. This is still an O(n) algorithm as each element in a collection will still need to be checked. An improvement here would be to create a new Java method as a URL resource that takes the ID of the client and gets that specific object back. This refactoring was not done because of time restrictions.

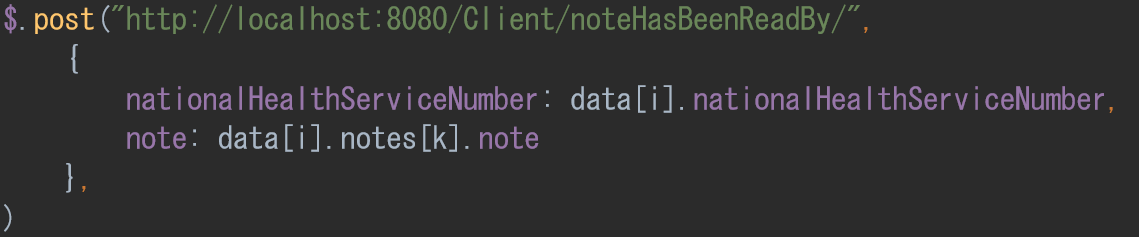
when a note is found it is checked against the hasBeenReadBy collection if it has not been read then a li tag is created and if it has been read then the li tag is also coloured grey.



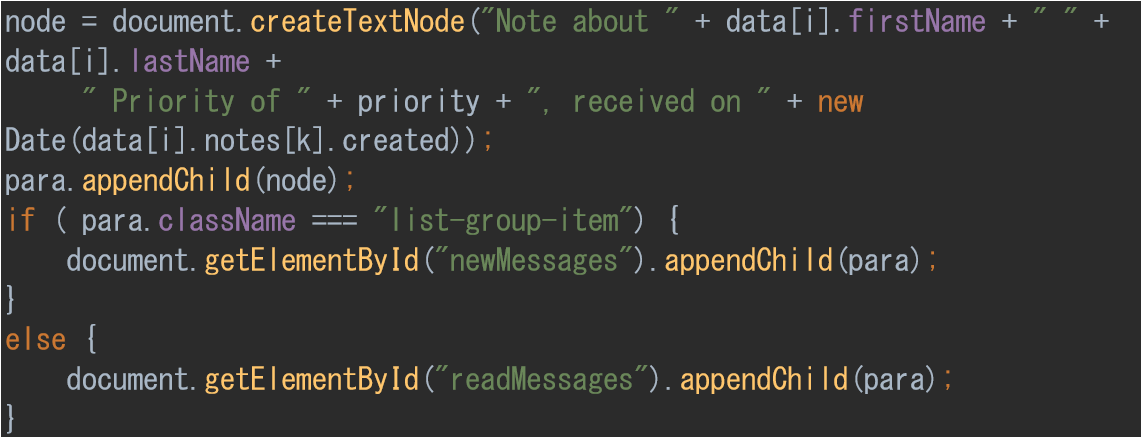
next the on click function is created for that li element.



This is created using a modal bootstrap function. And when clicked the hasBeenReadBy collect is updated.



when done, the new li element is added to the webpage, and this is repeated for each message and each supported client.



The last code needed was for the sort function; this works on an already populated list of messages. The sort algorithm is an altered version of one from w3schools.To test the system is functioning correctly, several tests where created.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Client | Support Worker | Supports client? | Can access Messages. | Notes. |
| 1 | Paul Smith | Tim Sims | yes | Yes | Messages are displayed and correctly coloured if not read. |
| 2 | Paul Smith | Kim Clark | yes | yes | Messages are displayed and correctly coloured if not read. |
| 3 | N/A | Mike Johnson | No | No | No messages display. Or options to create messages. |
| 4 | Clare Limes | N/A | No | no | No option to create messages for Clare lines. |
| 5 | Terry Stonham, Paul Smith | Tim Sims | yes | Yes | Can access message for both terry and Paul. |

With the completion of the note system tests, the system was shown to the charity. The charity accepted the note system with one desired change that the note would also store who created the note. A simple change by adding a ‘created’ field to the note class. This should have been caught earlier in development as the need to see who has sent a note is a common and essential feature. However, with the current time constraints, this change was added to the list of changes to be introduced after the September deadline.

### Part 5 - admin dashboard and advisory sub-system.

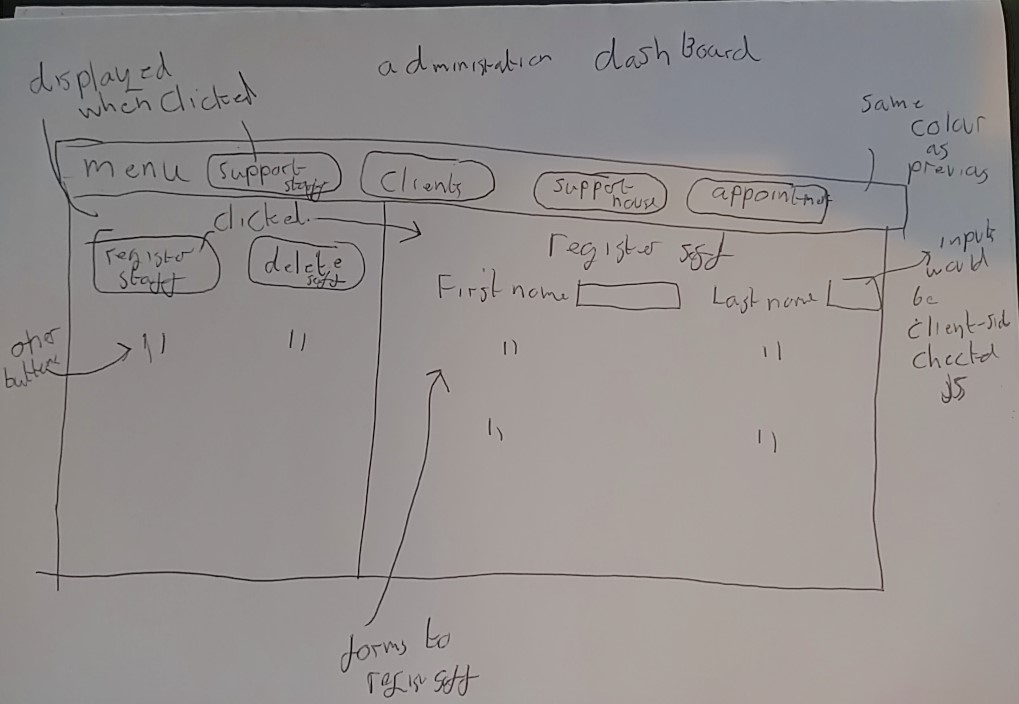
The design of the administration dashboard was to be like the already finish support staff’s and client’s dashboard.  (Fig 01 – original design of the administration dashboard.)

Figure one shows the original design made with the charity’s input. However, during development, some problems came to light. The most significate issue is where to put the program functions. For example. Removing an appointment from a client. Would that be part of the client buttons or appointment buttons; this could mean managers had to search for the functionality they need as most of the buttons were hidden until the group button was selected. In figure one, the register staff button is only visible after the support staff button is clicked.

One of the requirements picked up during the elicitation process was the need for managers to understand and quickly complete their tasks. If the managers struggled with the system and so did not accept it, the whole system would be pointless. Because of this, the design was dropped, and work began on a new design.

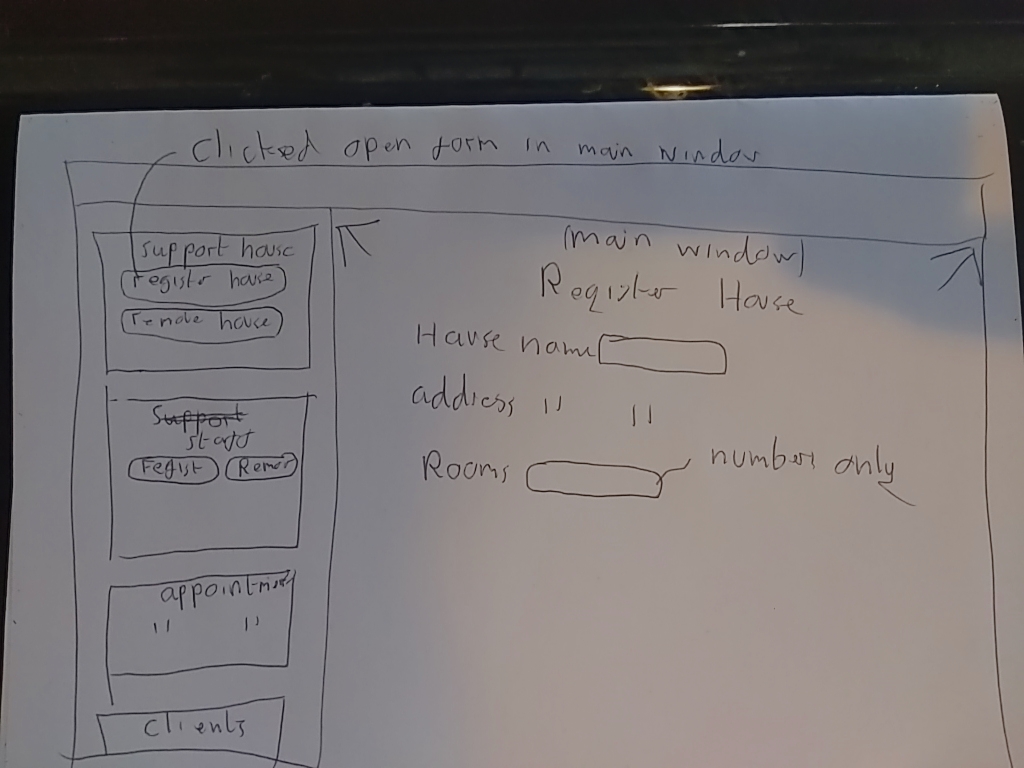
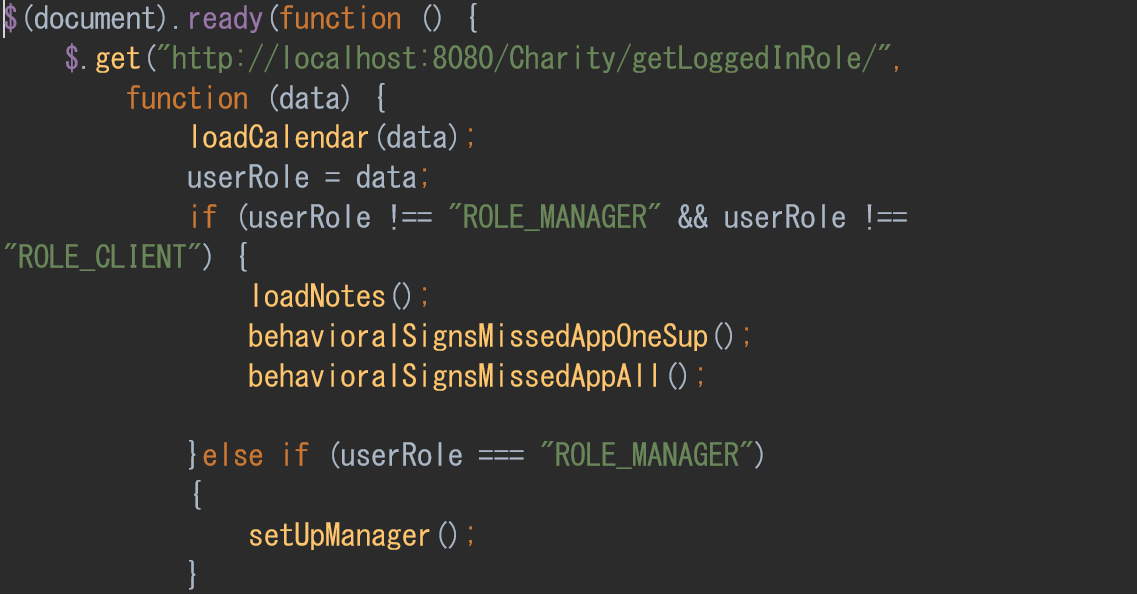
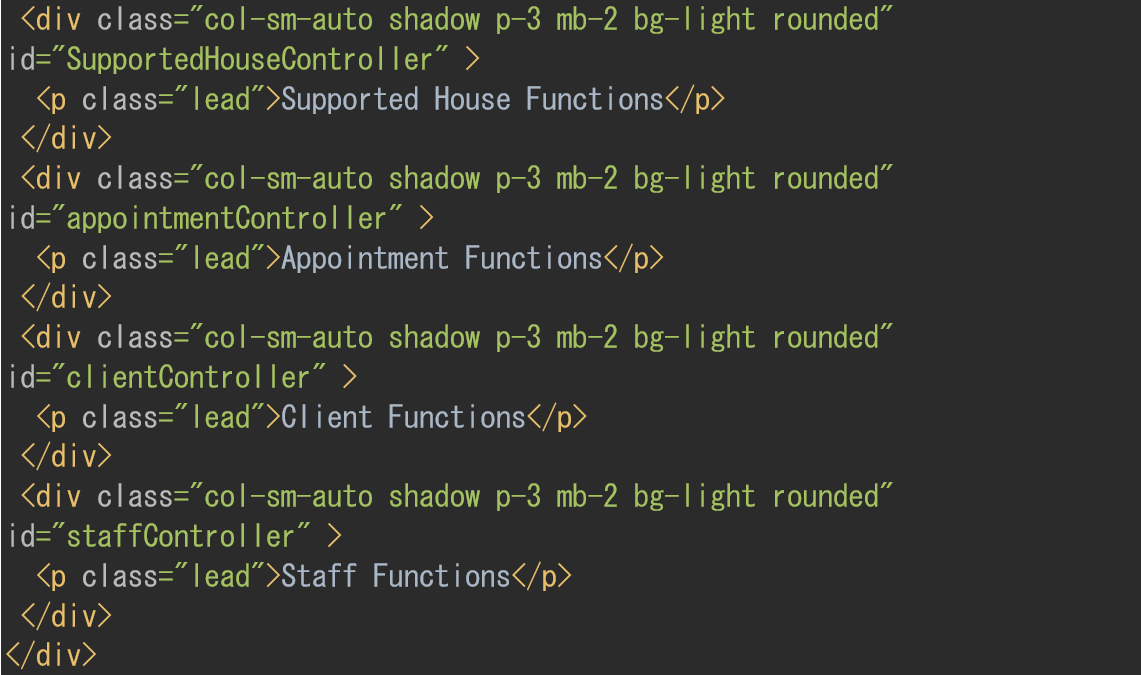
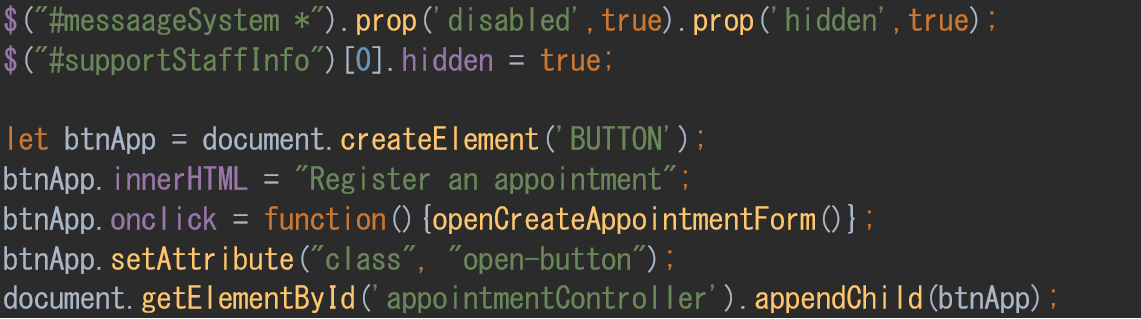
The idea for this new design was that most of the functions would be visible but still grouped. One of the differences between the design requirements for a manager and support staff member was the need for the latter to use a mobile device. Managers would be almost exclusively using a desktop with much more screen space.  (Fig 02 – the new design of the administration dashboard)

Figure two shows the new design that takes advantage of the space but is less usable on a smaller screen. To implement this, when the user logged-in based on their user-role the website would configure itself to that user.

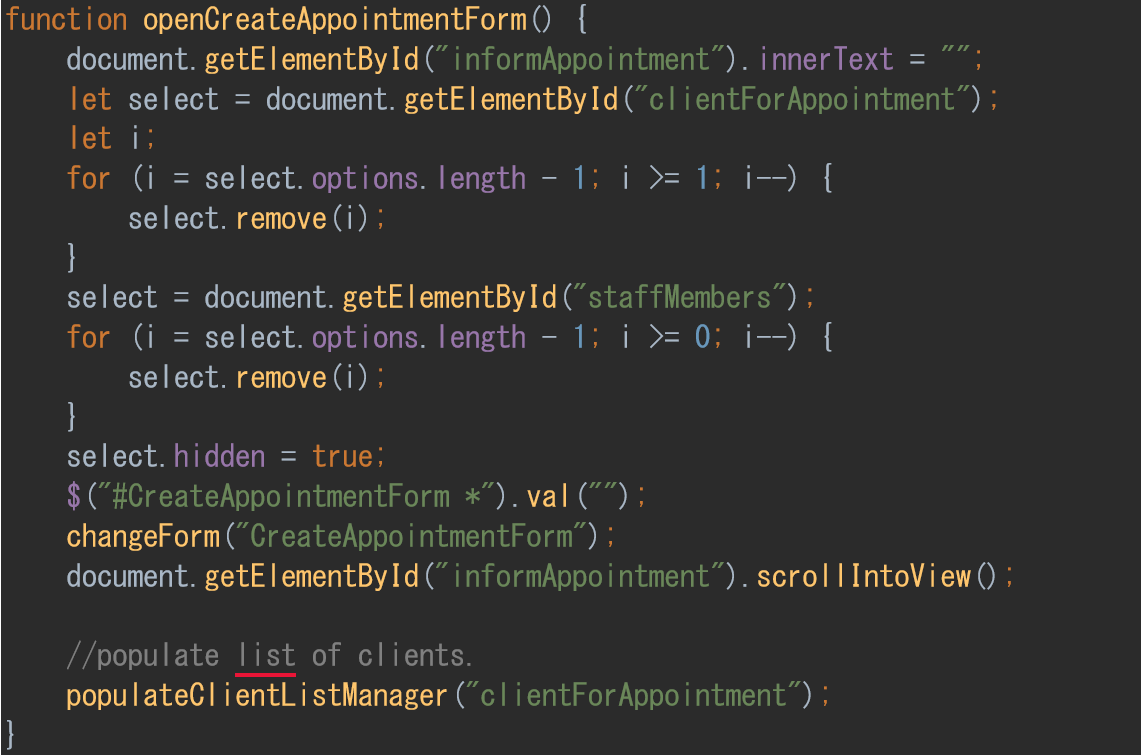


This code checks for the user-role and if they are a staff member loads the note and behavioural systems, and if the logging in user is a manager loads the management systems. This is split between the HTML and JavaScript

The HTML file has a collection of divs for each function set, and JavaScript is used to create the required buttons for each set.



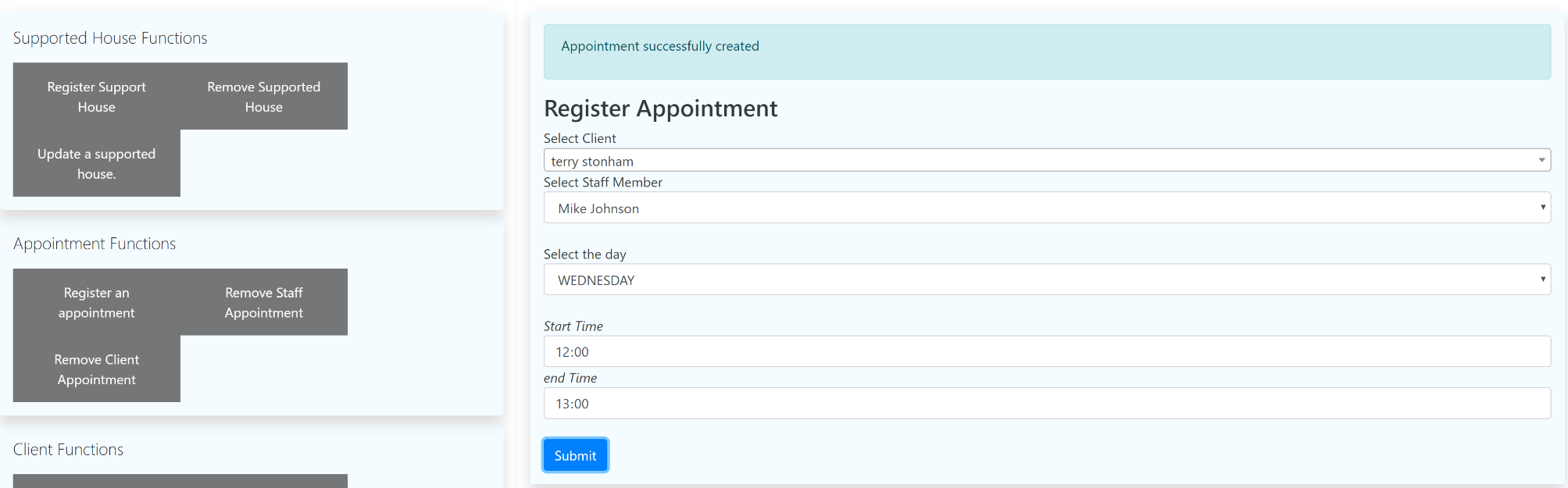
The message system is hidden, and a button with a function is created and appended to the div. Each button is similarly coded bit with different names. When the code is clicked. The function is then opened.

each button has a similar function that opens the form for that function and sets up its initial state.

The form itself is written in the HTML file and has an onsubmit function, the advisory system that is used to display information relevant to this form this can be anything from confirmations to errors. The rest of the form is a mix of labels and input boxes.

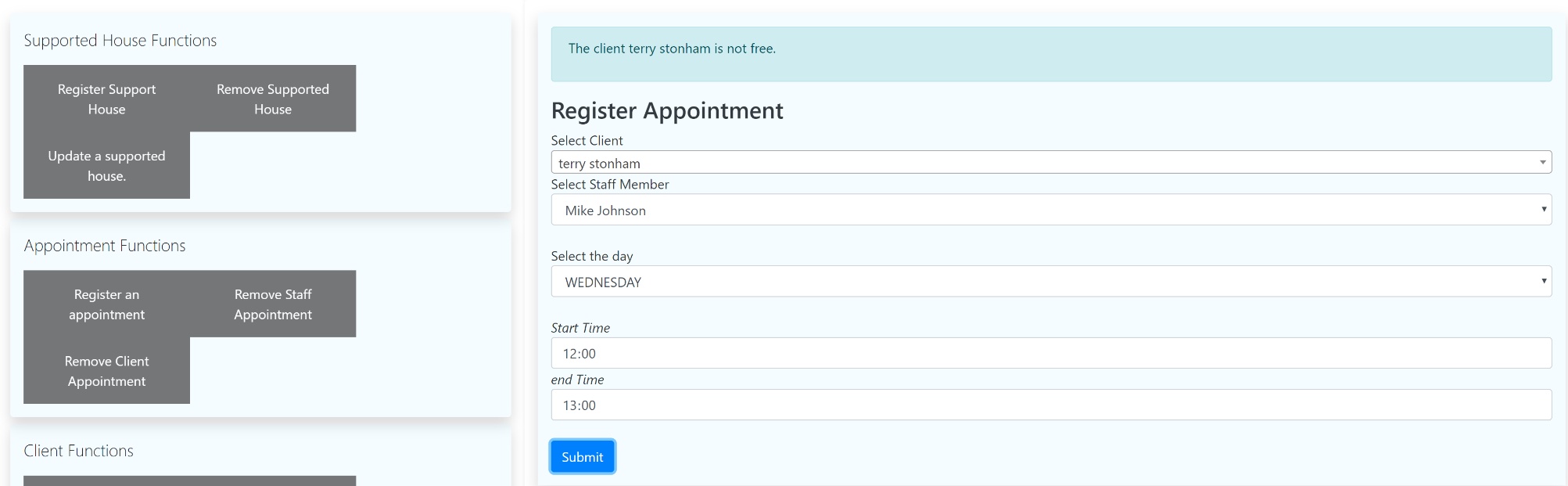


The submit function is what send the data to the backend Java and form that to the database. This code gets the information needed to create an accountment and sends that to the Java backend. Then a confirmation is returned and displayed using the advisory system, or if there was a problem, this is returned and displayed.



(fig 03 – registering an appointment)

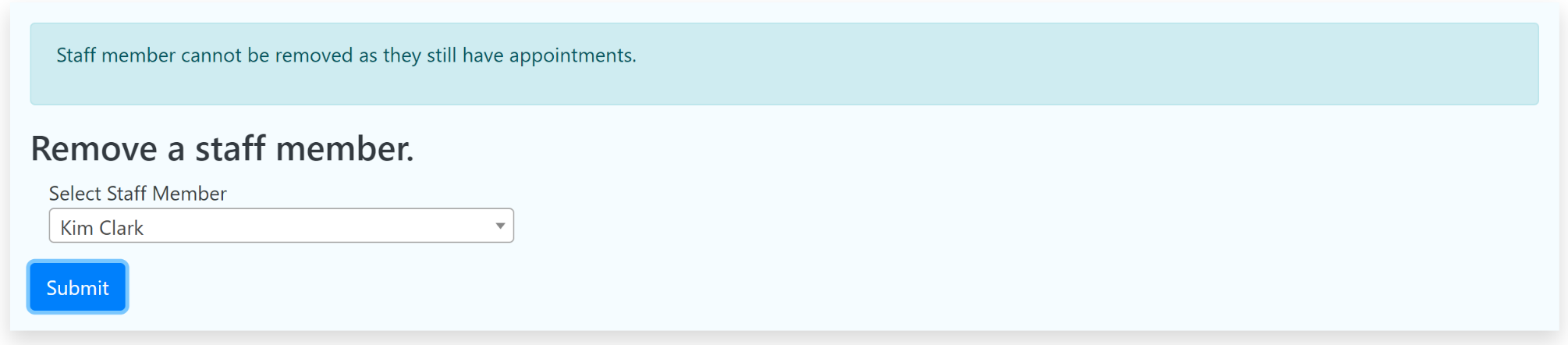
Figure three shows the page playout and an appointment being created between terry stonham and mike Johnson and a confirmation message displayed confirming the appointment. This form also helps minimise errors as only staff members with the correct specialisation for this client will be displayed.



(Fig 04 – registering an appointment error)

Figure four shows an error when the same appointment information is used again. The system informs the user that the client is not free at this time.

Another part of the advisory system is it will try and not just display the error but what can be done about it.

 (fig 05 – error removing a staff member)

Here figure five shows the advisory system showing an error and explaining why this staff member cannot be removed.

All the forms make use of error checking before submission to enforce design by contract, from simple checks of correct data format to more advanced checks. An example of this is registering a client.

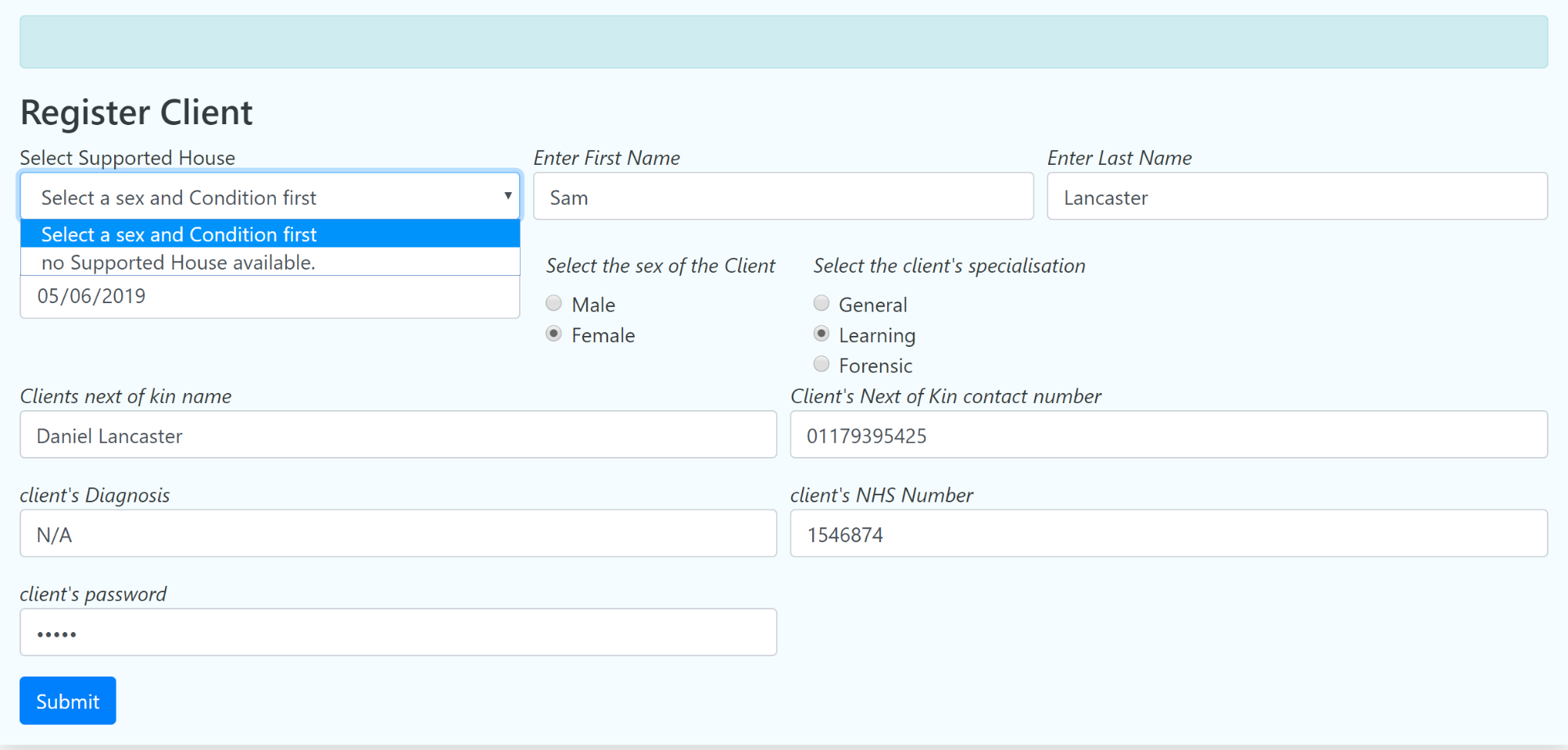
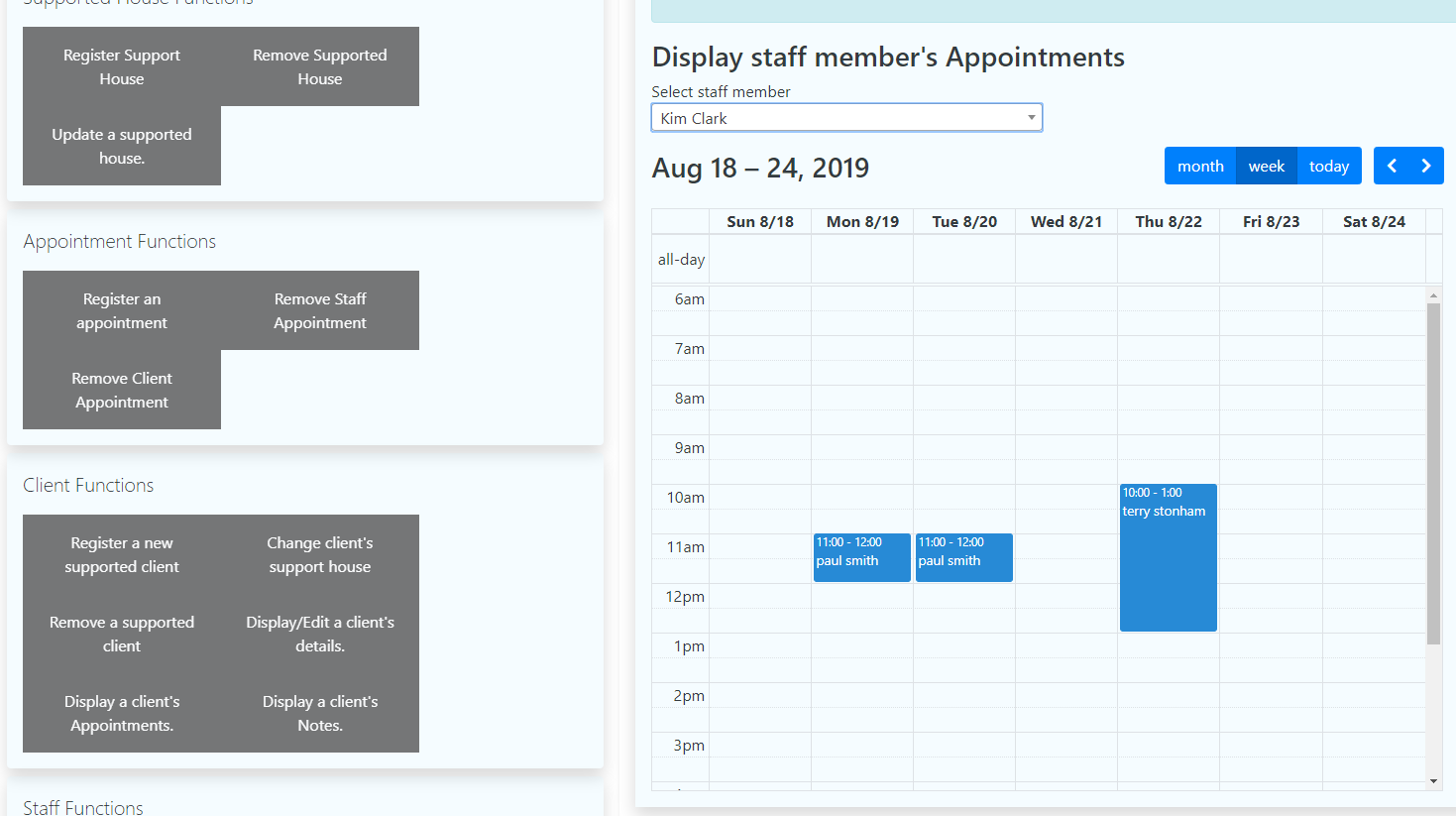
(Fig 06 – register client form)

Figure six shows that there are no available supported house’s with free rooms for this client, and so this client cannot be accepted until a room is available. But if this is needed, a temporary house can be created to house this client within the system.

One significate mistake made here was not handling the assignment of the key worker. It became evident during development that from the view of the system the key worker had no requirements over a support worker. From talking to the managers, it was agreed that the key worker would not need any specific system functionality and to the system, a key worker was the same as a support staff member. This distinction lead me to make a mistake, just because the system does not need to respond in any particular way to a key worker does not take away the need to model that relationship. So, when developing the system, I did not create the functionality to associate a support staff member as a client's key worker even though this was part of the requirements document. The ability to specify a key worker for a client has been added to the waiting room as a priority.



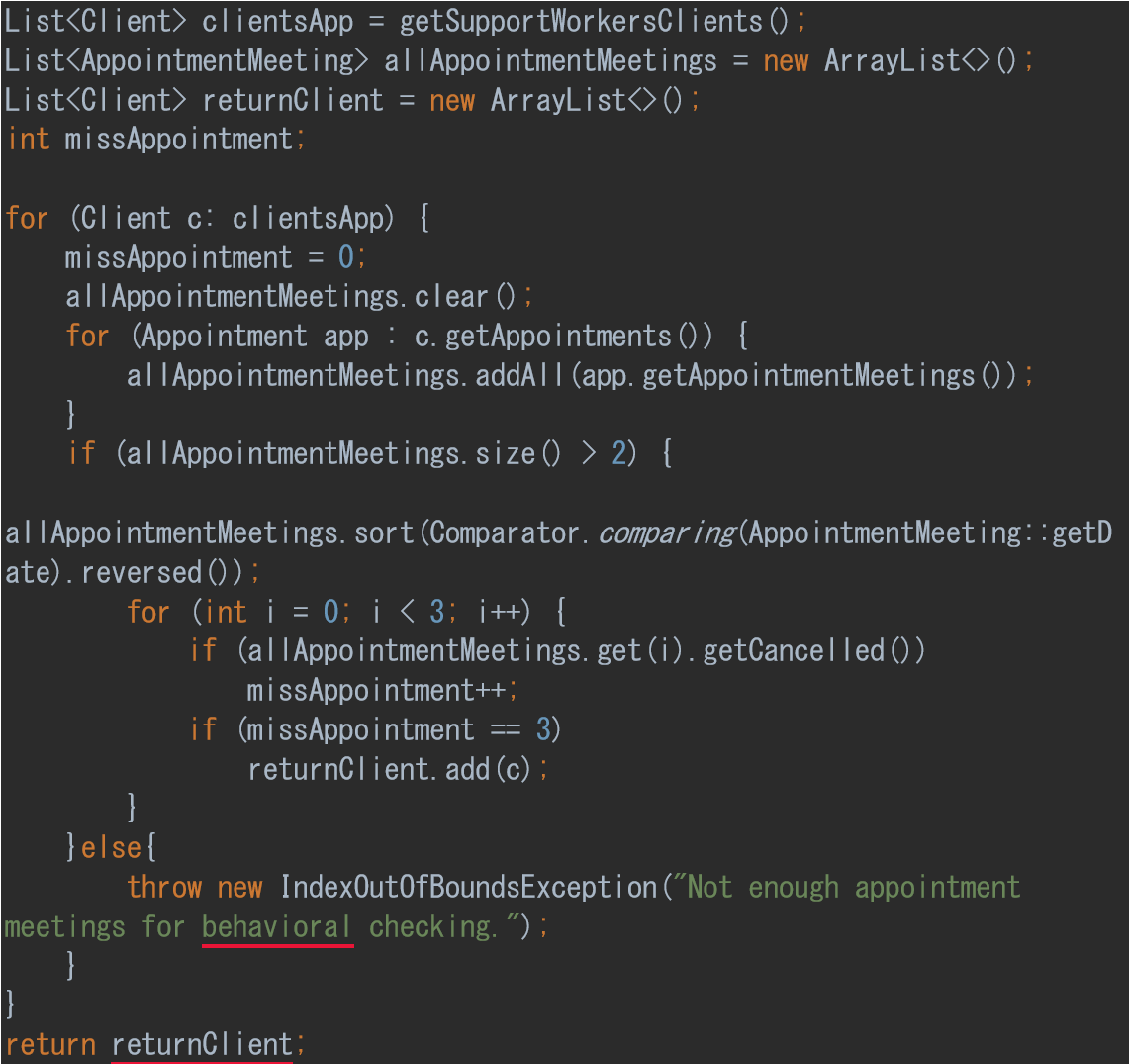
(Fig 07 – manager accessing the staff member Kim Clark’s appointments)

Figure seven shows the managers ability to access the calendar of support staff members. This calendar has the same functionality as the support staff members.

### Part 6 - behavioural flagging and advisory sub-system.

The behavioural flagging system relates to UC24 on page 36 of appendix A. This can be broken down into two requirements first if a client has cancelled three appointments in a row second if they have cancelled three appointments with a specific support worker.

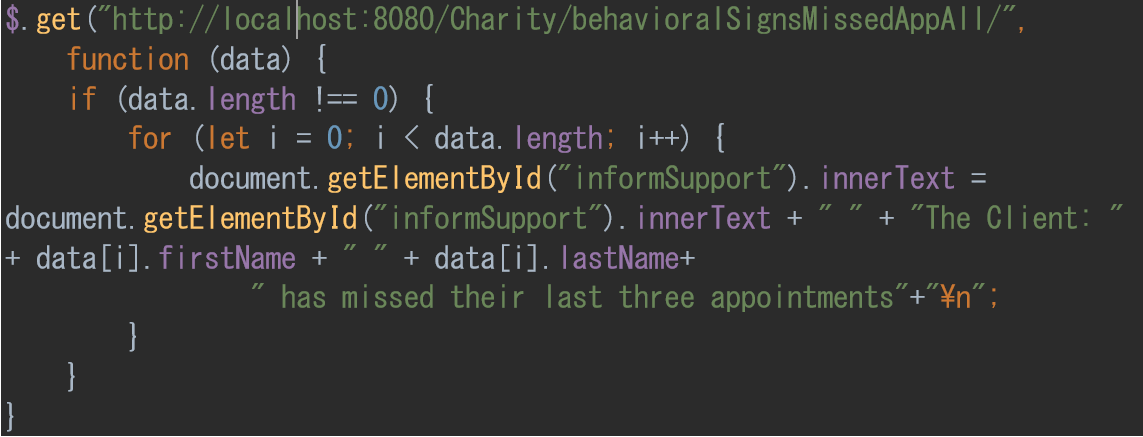
One change from the use case was instead of using the message system the warning would be displayed on the advisory display this would mean the information is almost constantly displayed to the user not hidden in a note.

The system is written in two methods in the CharityController class the first method is related to three missed appointments irrespective of a support worker.

This method checks each client supported by the logged-in user for cancelled appointments if there have been three in a row then that client is added to an array and returned to the front-end for JavaScript to make use of.

The method for checking for specific cancellations is like the previous method but checks appointments instead, if an appointmentMeeting has three cancellations in a row that appointment is added to a LinkedList and return to the front-end.

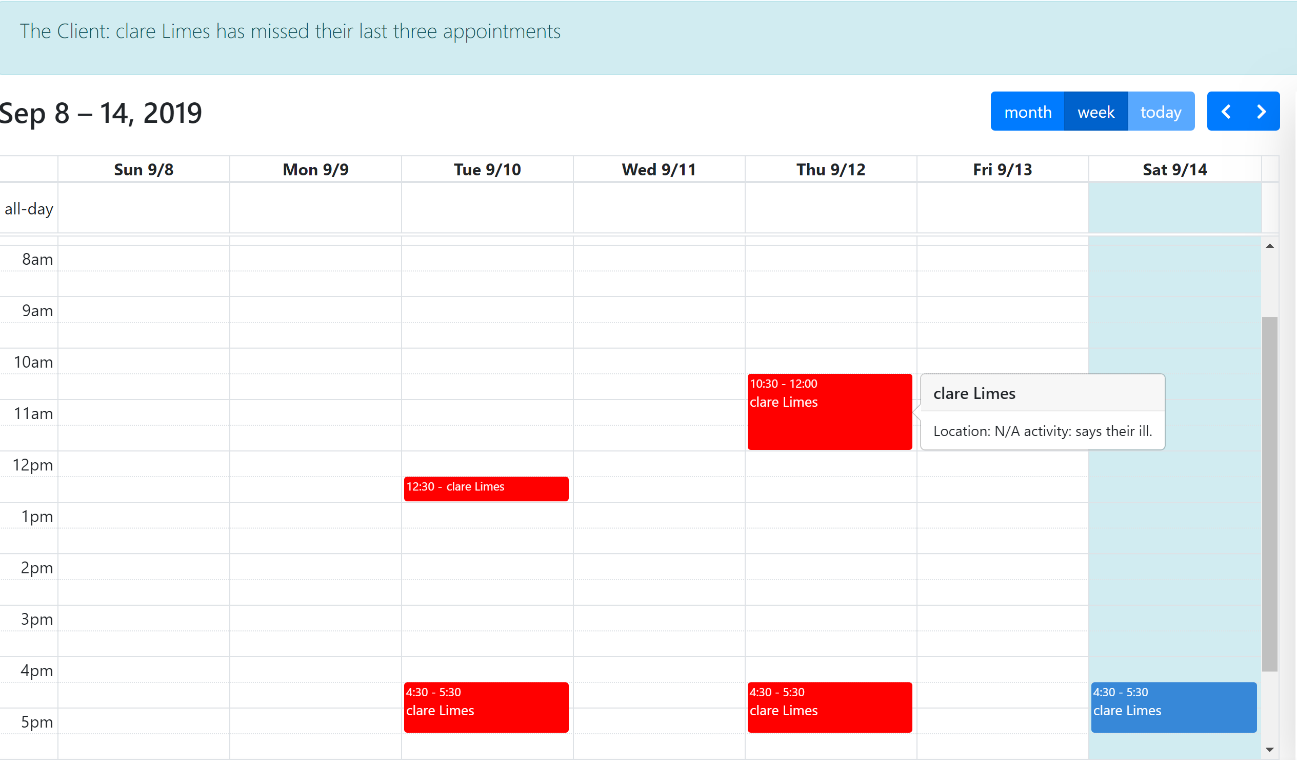
The JavaScript functions for both methods are very similar.



The return collection is check for data and if so, iterated over with the message displayed to the advisory display.

To test these methods.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Client | Support Worker | Supports client? | Missed three appointments | Missed three appointments with specific staff | notes |
| 1 | Paul Smith | Tim Sims | yes | Yes | no | advisory displayed a message about miss appointments |
| 2 | Paul Smith | Kim Clark | yes | No | Yes, with Tim sims | advisory displayed a message about miss appointments with Tim sim |
| 3 | Clare Limes | N/A | no | yes | yes | No message displayed about Clare limes |
| 4 | Terry Stonham | Tim Sims | yes | no | No | No message displayed about Tim Sims |
| 6 | Terry Stonham,  Paul Smith | Peter Keeper | Yes | Yes  Paul Smith | no | advisory displayed a message about miss appointments with paul Smith |



(Fig 08 – the support staffs screen)

Figure eight shows the advisory system displaying that Clare Limes has missed their last three appointments. It also shows one of the missed appointments being clicked and the message “says their ill” as the reason given for the cancelled appointment.

The behavioural flagging system has the most potential for improvement, not in its current implementation but in expanding its scope. An example of this that was discussed with the charity was an imaginary client that consistently cancelled appointments around a specific date or if the charity were to implement a mood monitoring system a consistent drop in mood around one particular date, the same period each year, this kind of information would be useful to a therapist or community psychiatric nurse as an area to investigate. While this is an exciting idea, it is outside the scope of the current system but worthy of future studying.

This is the end of the project development concerning TM470 at this point, most of the use cases from the requirements document have been met, but there has been some deviation in design from that document.

## Review of current stage of project work

The project is mostly feature complete; some of the appointment time systems are not working correctly. Currently, the system adds 15 minutes for each appointment as their time needed to travel. I still would like to implement a better system that works out the actual travel time between appointments.

I have thought about making use of something like google maps for this using the postcodes of each house or location to workout travel time, but this would probably have a financial cost and most likely take a few weeks to implement.

Another area that is not functioning is around allowing Managers to access the mail system of individual staff. This was a question of time, I put the work off to finish the behavioural system as this was more important to finish and ran out of time. I do plan to correct this and do not see it taking more than a few days.

When these problems are fixed the plan is to use the system shadowing the current paper-system for several weeks, this will hopefully find any bugs in the website and ensure the software can handle the same situations as the paper system.

## Review project management

At this point, I can say that the use of an incremental and iterative waterfall approach with test-driven development has made this project manageable. Breaking down the program into a collection of subsystems that are in a few cases, self-contained has helped control the overall complexity and meant I have always had an idea of what I should be doing and where I should be in the whole project. Even integrating iterations was painless because of the collection of JUnit tests I had available for regression testing.

However, there were several problems as well with this approach. Firstly, one of my reasons for using an iterative waterfall approach was to add the needed development research before the iteration that would use it this choice was almost a disaster. When the time came to implement Spring Framework I had completely underestimated the work and time needed. In the future, I would perform some In-depth research of the technologies each iteration would use. This did though show strength of an incremental approach as implementing spring framework was a much more manageable task on the second increment versus if this was a standard waterfall approach and half the system ended up needing altering. The ability to see mistakes early and correct them seems an advantage of the incremental and iterative waterfall approach.

### Resources, skills and methods

#### Resources

* IntelliJ IDE

IntelliJ is an integrated development environment that I used for the entire project. I had no problems procuring it and have used it in the past.

* [www.draw.io](http://www.draw.io)

I used draw.io for all of the projects models and diagrams. I had already used this product during TM354, so was familiar with it. One of the potential problems with it was the need for an internet connection. However, I covered this by downloading the offline version that I did not need in the end.

* Stakeholders, other staff at the charity

The team at the charity where crucial to the project. The project in its current state would not exist without their input, especially around the manager dashboard. Overall I did have some reservations about overloading staff with questions and designs, but in the hole they were enthusiastic.

* stackoverflow.com

stackoverflow is an online developer community; this was vital to the project. I used the forums extensively when programming the project. but the problem I had with spring framework and JSON conversion I would most likely not have fixed without a forum thread on the same issue.

* Chrome’s DevTools

A website development toolset which will allow debugging of HTML. This tool allowed me to debug my javascript with a webpage that was online. Also, to check the functionality of the responsive design for the support staff functions.

* Junit

Junit is a unit testing framework that allows for better regression testing and debugging. Junit was vital to the project the ability to regression test between iterations helped catch bugs and build confidence in the successful integration of new iterations.

##### Methods

* object-oriented design

Object-Oriented design is a design philosophy where software is created modelling the real world. OOM is not the only design philosophy, but the only one I have any in-depth knowledge of. So, it is hard for me to say the impact this choice had on the project without experience in other design philosophies.

* Volere template

I used the Volere template for the basis of the project in some ways; it is an essential part of it. The document itself is similar to the TM354 software engineering case study but more in-depth. Like with OOM there are other design philosophies versions of big A agile for example, but I do not know them so using Volere made sense.

* multi-layered architecture

An n-tier architecture was used for the project. With java acting as the back-end, spring framework as the middleware and javascript and the HTML as the front-end this worked well and fit in with an incremental approach.

##### Skills

* Java

What was taught by the OU and the primary language I have experience with. Was used for all the backend development. There are no real risks to using java. It is a popular language with a robust library and has an active and helpful community.

* HTML and CSS

HTML (Hypertext Mark-up Language) is the standard for creating the structure of websites. CSS (Cascading Style Sheets) is used for controlling the style of most websites.

* UI Design

I did not end up using wireframes for my plan because of time constraints, but I still made use of functional prototypes and prototype throwaways in the shape of pen and paper designs.

* restful architecture

with the project complete I would say that the Spring framework I used modelled remote procedural calls more than restful architecture this again was a question of time and the mistake I made introducing spring framework so late with insufficient research.

* JavaScript and JSON

I used JSON exclusively for sending and receiving data between the front-end and back-end. JSON did work well but did create some problems with the conversion between Java and Javascript. JavaScript was used for the front-end to format and display data sent from the back-end. JavaScript was a risk as I had less knowledge of it than I thought I had. However, through books and searching, it did not become a problem.

* Database design

I did not end-up designing a database, I instead used Spring Framework to manage all database work this was problematic as I ran into many problems with its implementation and a few I could not have fix without using stackoverflow. In hindsight designing a database may have been a better choice.

## Review of personal development

Having reached the end of this project, I think one of the core skills I have been forced to develop has been learning to be critical of my work but also having the self-belief that I could recover and improve from my mistakes.

However, two areas that have caused me continual problems have been underestimating the work involved in each iteration this is related to the second problem that has happened several times. Not giving the requisite amount of time necessary to understand the technology involved with that iteration this has meant that on multiple occasions I ended up making significant changes to my code, for example, the Charity class is entirely different to its original inception.

My mistake with Spring Framework was again underestimating the needed time to understand Spring and so, beginning work on the first Iteration documentation and code before realising the extent of the changes I would need to introduce. That put me in the position of abandoning Spring and continuing with the system I had already finished. Alternatively, investing the time required to alter my code to incorporate Spring Framework and trust that I could learn the Spring functionality I needed and stay on schedule.

After wasting about a week on this decision, I decided to use Spring. I think this was the correct choice it was challenging but allowed me to produce a better program in a shorter time. This experience helped me learn to code what I was doing at the moment but have an eye on want was coming. Making sure I was reading the right documents and always had some idea of what was needed next even if it was just an outline.

I believe that this could have been entirely avoided by spending more time at the start of the project reading and researching the implementation of the technologies I had no experience with like Spring framework. In hindsight incorporating Spring into the first iteration of the basic object, models would have corrected most of this and meant the first iteration would have stayed more relevant as the project went on.

This experience has not just helped with my Java programming skills, but I have learned to skim technical documents to find the parts I needed. Judging the text or information source was relevant in the first place and if so, investing the time to give a more in-depth read. This skill of synthesis has been a core aspect of this project especially with my choice to use Spring, and MongoDB two technologies I had no experience with so, required me to learn them at the same time I was implementing them.

Another area that has improved throughout this project has been time management. Attempting both TM470 and TM352 at the same time was challenging. I initially tried dividing my time between the two based on what was the priority for that time TMA’s etc. but as deadlines drawled in, I consistently stop sharing my time and spending all of it on completing the TMA. After that was complete, I found I was behind in the neglected course, and this would repeat for each TMA and became a real problem for TM352’S EMA. One way I tried to improve this was to set aside two hours each evening for the neglected course, especially during the EMAs.

Looking back, one of the most daunting but fulfilling aspects of this project has been the need to be self-reliant even with a tutor, the only expert on your project is yourself. Where in the other units the work was always a small part of someone else’s design. Needing to follow the life cycle of my project from conception to delivery has solidified what it means to work on a software project and the kind of work I will have to do in a real working environment. Moreover, to now know it is within my capability to deliver that.

## References

Lieberherr, K. and Holland, I. (1989). Assuring good style for object-oriented programs. *IEEE Software*, 6(5), pp.38-48.

Larman, C. (2005). *Applying UML and patterns*. Upper Saddle River (New Jersey): Prentice-Hall.

Information Commissioner’s Office (2015), ‘Assessing Adequacy International data transfers ‘[online]. published 29 September 2015, Available at https://learn2.open.ac.uk/pluginfile.php/2422879/mod\_resource/content/1/assessing\_adequacy\_international\_data\_transfers.pdf (accessed 11 January 2019)

Robertson, J. and Robertson, S. (2006). *Mastering the requirements process*. Harlow: Addison-Wesley.

Walls, C. (2016). *Spring Boot in action*. Shelter Island, NY: Manning Publications.

Neward, T. (2005). *Effective Java*. Boston: Addison-Wesley.

Freeman, E. and Freeman, E. (2005). *Head first design patterns*. [Sebastopol, CA]: O'Reilly Media.

Garcia, A. and Farcic, V. (2018). *Test-driven Java development*. 2nd ed.

Achecker.ca. (2019). *IDI Web Accessibility Checker : Web Accessibility Checker*. [online] Available at: https://achecker.ca/checker/index.php [Accessed 15 August. 2019].

Information Commissioner’s Office (2014), ‘Data controllers and data processors:

what the difference is and what the governance implications are’ [online]. published 6 May 2014, Available at https://learn2.open.ac.uk/pluginfile.php/2422880/mod\_resource/content/1/data-controllers-and-data-processors-dp-guidance.pdf (accessed 11 January 2019)

postman. (2019). Postman, Inc. Available at <http://www.getpostman.com>

Stack Overflow (2017) [Spring-data-mongodb @DBRef causing infinite recursion? I can't work out why](https://stackoverflow.com/questions/44038786/spring-data-mongodb-dbref-causing-infinite-recursion-i-cant-work-out-why) - Stack Overflow [Online]. Available at <https://stackoverflow.com/questions/44038786/spring-data-mongodb-dbref-causing-infinite-recursion-i-cant-work-out-why> (Accessed 7 July 2019).

fullCalendar. (2019). FullCalendar LLC. Available at <https://fullcalendar.io/>

Mark Otto, a. (2019). *Modal*. [online] Getbootstrap.com. Available at: https://getbootstrap.com/docs/4.0/components/modal/ [Accessed 12 July. 2019].

W3schools.com. (2019). *How To Sort a List*. [online] Available at: https://www.w3schools.com/howto/howto\_js\_sort\_list.asp [Accessed 15 July. 2019].

Baeldung. (2019). *Java, Spring and Web Development tutorials*. [online] Available at: https://www.baeldung.com [Accessed 12 July 2019].

# appendix A

Developing a web application to facilitate time management and communication between support staff members.

Contents

[Project Drivers 2](#_Toc3373058)

[1. The purpose of the project. 2](#_Toc3373059)

[Goals of the Project 2](#_Toc3373060)

[2. The Stakeholders 3](#_Toc3373061)

[Users: 3](#_Toc3373062)

[Project Constraints 5](#_Toc3373063)

[3. Mandated Constraints 5](#_Toc3373064)

[4. Naming Conventions and Terminology 5](#_Toc3373065)

[5. Relevant Facts and Assumptions 6](#_Toc3373066)

[Relevant Facts 6](#_Toc3373067)

[Business Rules 6](#_Toc3373068)

[Assumptions 6](#_Toc3373069)

[Functional Requirements 7](#_Toc3373070)

[6. The Scope of the Work 7](#_Toc3373071)

[The Current Situation 7](#_Toc3373072)

[Specify Business Use Cases 10](#_Toc3373073)

[7. The Scope of the Product 16](#_Toc3373079)

[Product Use Case Table 19](#_Toc3373081)

[Individual Product Use Cases 20](#_Toc3373082)

[8. Functional Requirements 37](#_Toc3373083)

[Non-functional Requirements 53](#_Toc3373181)

[9. Look and feel requirements 53](#_Toc3373182)

[10. Usability and humanity requirements. 53](#_Toc3373183)

[11. Performance requirements. 54](#_Toc3373184)

[12. Operational and environmental requirements. 54](#_Toc3373185)

[13. Maintainability and support requirements. 54](#_Toc3373186)

[14. Security requirements. 54](#_Toc3373187)

[15. Cultural requirements. 55](#_Toc3373188)

[16. Legal requirements 55](#_Toc3373189)

[Project issues 56](#_Toc3373190)

[17. Open Issues 56](#_Toc3373191)

[18. Off-the-shelf Solutions 56](#_Toc3373192)

[19. New problems 56](#_Toc3373193)

[20. Tasks 56](#_Toc3373194)

[Planning of the Development Phases 58](#_Toc3373195)

[21. User documentation and training 59](#_Toc3373196)

[22. Waiting room 59](#_Toc3373197)

# Project Drivers

## The purpose of the project.

A charity runs a community-supported living project in the UK. And plan to replace their current paper-based scheduling and management system with a website.

This new system will allow managers to add or remove clients, support staff members and supported houses. To setup timetables and access appointment meeting notes. The manager will also have the facilities to create and change client support teams.

support staff members will be able to access their timetables. Their individual timetables will display which client they are to see and what time the appointment is for. They will be able to record when they started the support session, the support session ended and what was done. support staff members will also be able to send and receive messages related to a client they support and specify the importance of the message.

The system will handle potential problems like a client being assigned two or more key workers and timetable conflicts based around travel time and appointment availability between a client and support staff member. The system will also flag several set behavioural patterns e.g. a client is continually cancelling support or a client cancelling the support of a specific support staff member.

### Goals of the Project

#### Purpose:

To speed up and automate scheduling and improve messaging between support staff members.

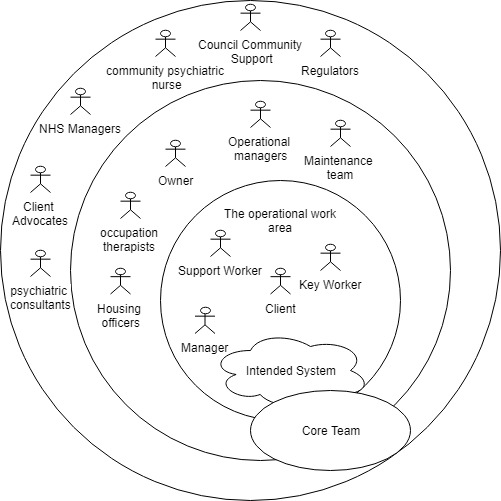
#### Advantage:

To cut down amount of time spend on paperwork. And improve communication between support staff members.

#### Measurement:

Reduction in support hours spent on paperwork by 50 percent. Improve response time to messages and alerts by 75 percent.

## The Stakeholders



### Users:

* **Managers** - Key users of the system.
  + User role

Manage assignment of support staff members, choosing key workers. creating timetables and managing the supported houses.

* + Subject-matter experience:

Managers will have extensive experience in mental health support.

* + Technological experience:

Novice.

* + Intellectual abilities:

Managers are generally experience and intelligent. Mental health care is constantly changing and evolving.

* + Attitude toward the job:

Managers will have years of experience in mental health. Which can be a stressful and difficult occupation. Most have a dedication to helping and supporting others.

* + What is it about their work that they most wish to improve?

Most managers wish to spend less time on paperwork. Especially as time on paperwork is counted as support hours for the client the paperwork relates too.

* **Support staff member** - Key users of the system.
  + User role:

Provide support to clients on their timetable. Monitor clients for deterioration in mental health and if necessary, report to key-worker or on-call.

* + Subject-matter experience:

support staff members can be relatively new and inexperience in mental health support.

Technological experience:

Novice.

* + Intellectual abilities:

support staff members do not require any specific levels of education. And can come from all back-grounds.

* + Attitude toward the job:

support staff members are generally new to metal health. Which can be a stressful and difficult occupation. It is common to have a high turnover of support staff members.

* + What is it about their work that they most wish to improve?

Most support staff members wish to spend less time on paperwork. Especially as time on paperwork is counted as support hours for the client the paperwork relates too. And most support staff members wish from a better system for handling information exchange between appointments.

* **Key Worker** - Key users of the system.
  + User role:

Manage a specific clients mental health. Be their main point of contact for other metal health services. And our responsible for their medication.

* + Subject-matter experience:

A Key Worker will have experience in mental health support. And training in specific mental health conditions.

* + Technological experience:

Novice.

* + Intellectual abilities:

Key Worker are generally experience and intelligent. Mental health care is constantly changing and evolving. And Key Workers are expected to stay aware of current mental health programs.

* + Attitude toward the job:

Key Workers are generally experienced and dedicated. Working in the metal health system can be a stressful and difficult occupation.

* + what is it about their work that they most wish to improve?

Most Key Workers wish to spend less time on paperwork. Especially as time on paperwork is counted as support hours for the client the paperwork relates too. Key Workers also spend a lot of time chasing up on information about the client they are responsible for from other support staff members.

* **Client** - Secondary user of the system.
  + Subject-matter experience:

Will not necessarily have any metal health knowledge or experience.

* + Technological experience:

novice.

* + Intellectual abilities:

Clients can come from all levels of education and intellectual capability.

* + What will the system improve for the user?

The ability for clients to access a timetable will allow them to stay aware of who is to see them and when. Also, the ability to access who their support staff members and key worker are and get contact information.

# Project Constraints

## Mandated Constraints

* Description: Completed by the 16 of August 2019.

Rationale: when the project must be completed with time to write up the EMA

* Description: The system will be accessible by the last five updates of Chrome, Firefox, Safari and Edge. Desktop and mobile versions.

Rationale: currently the most commonly used browsers.

Supporting Materials: Browser Statistics [WWW Document], n.d. URL <https://www.w3schools.com/browsers/> (accessed 2.11.19).

Fit Criterion: An agreed browser compatibility testing suit.

## Naming Conventions and Terminology

**Client**: a person with a mental health condition that has been given a care package by their local council.

**Timetable**: a collection of papers that hold the appointment times for a support staff member to meet and support a client. Is used to record when the support staff members arrived at the appointment what they did during the support session and when the support session finished and any observations. if the client rejected the support session then the reason is recorded. At the end of each week the timetable is given to a manager.

**Support Worker**: provides support to a client and monitors for deterioration.

**Key Worker**: takes responsibility for a specific client. Acts as a contact between NHS mental health teams and community support charity. A key worker can also act as a support worker for other clients. But not all support worker are key workers.

**Support Staff**. Used to refer to support workers or key workers.

**Manager**: manages timetables, supported houses, support staff members. And contact with local council.

**local council**. Local government that controls financing of support hours. Requires documentation of hours provided and how the time is in-line with the clients care plan.

**local council care plan.** this is the document created when a client is assessed and given support hours. It contains the reasons why the care hours where supplied and how they should be used.

**support charity care plan**. a plan of care related to a client created between the client the clients HNS mental health team and the support charity. The client may also have a care plan within the HNS mental health team that is separate from the support charity.

**Support team**. The support staff members and key worker assigned to support a specific client.

**support session.** The support hours spent supporting a client. a support staff member or a key worker provides this time.

**Support Hours.** The amount of time provided to a client per week. This includes admin hours and supported hours.

**forensic mental health**. Supporting people with mental health disorders and a history of criminal offending. Only trained Support Staff provide this support.

**General mental health**.Supporting people with general mental health conditions. No specific training required outside of experience with a specific condition.

**Learning Difficulty.** Supporting people with learning difficulties. Depending on the Learning difficulty specific training can be required.

## Relevant Facts and Assumptions

### Relevant Facts

* No more than 100 users will be connected to the system at any one time.

### Business Rules

* Support hours can range from 30 minutes to 2 hours.
* Supported houses are safety checked once per week.
* A client can only have one key worker.
* A support staff member can only act as a key worker for at most 3 clients.
* A client can only be accepted if there is a free room in a supported house.

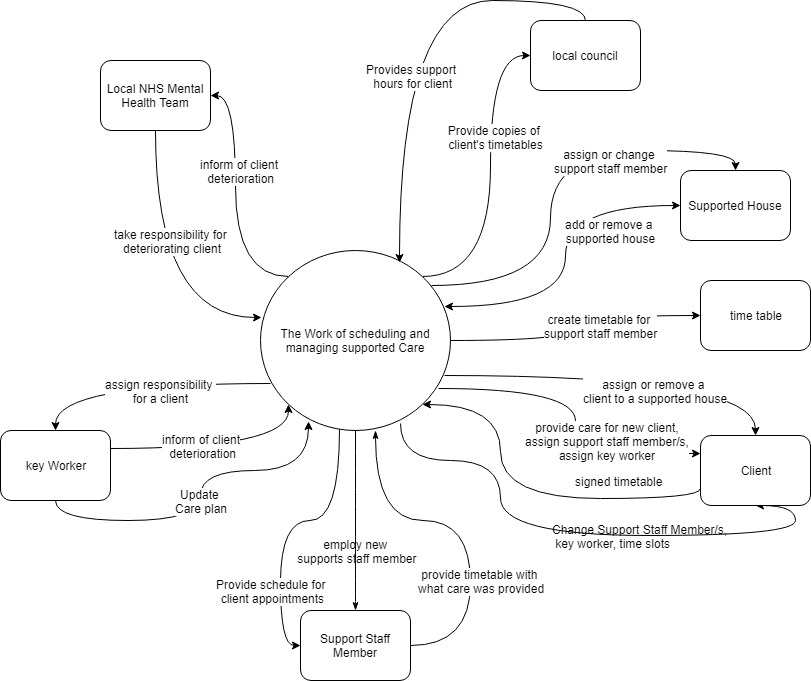
### Assumptions

* Clients will have access to the internet to make use of this system.
* All support staff will have modern mobile devices.
* All support staff walk between appointments.

# Functional Requirements

## The Scope of the Work

### The Current Situation

****

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Event Name** | **Input and output** | **Summary of BUS** |
| 1 | local council Provides support hours for client | Client care overall hours (in) | Record the hours a client has been given and the details of their needs. |
| 2 | Local Council receives signed timetables. | copies of client's timetables (out) | Provide copies of client's timetables to the local council. |
| 3 | Supported House assigned new support worker. | support worker assigns to a supported house (out) | Record the new support worker assigned to the supported house. |
| 4 | Add or remove Supported house. | New supported house added or removed (in/out) | Record details of the new supported house or remove details. |
| 5 | Create a timetable for a support staff member. | Print the timetable of a specific support staff member. (out) | Print a list of all clients the support staff member is meant to meet that week. With what day and time. |
| 6 | assign or remove a  client to a supported house | Record which supported house a client will be living. (in) | Record which supported house a client will be living is will be used for support worker meeting location. |
| 7 | provide care for a new client,  assign support staff member /s, assign the key worker | Record a new client and assign their support team. (out) | A new client is assigned a support team based on their needs. The support hours are broken down between this team. |
| 8 | Client signs timetable | A signed timetable. (in) | The client signs the timetable to show that he agrees with the time support started and ended and what they did for that time. |
| 9 | Change support staff member /s, key worker, time slots | Change members of a client’s support team. (out) | Make changes to the support team around a client or the client's support times. |
| 10 | a support staff member provides timetable with  what care was provided | A signed timetable. (in) | A signed timetable with the time's support started and ended and what the support staff member did with the clients for that time. |
| 11 | employ a new  support staff member | Record a new support staff member. (out) | A new support staff member is employed. |
| 12 | Provide a schedule for  client appointments | Provide a printed schedule for  client appointments. (out) | a printed schedule for  client appointments that week. |
| 13 | Keyworker updates Care plan | An updated care plan. (in) | An updated care plan for a specific client with new goals and needs. |
| 14 | Keyworker informs of client deterioration. | Record concerns about a specific client. (in) | The Keyworkerinforms of their concern related to a client they support. |
| 15 | A support staff member is assigned responsibility for a client as their key worker. | Record a specific support staff member as a client is one key worker. (out) | A support staff member is selected as a client’s key worker. |
| 16 | NHS mental health team take responsibility for deteriorating client | A record that an NHS mental health team has taken responsibility for deteriorating client | an NHS mental health team has taken responsibility for the deteriorating client. Through intensive team or sectioning. |
| 17 | inform NHS mental health team of client deterioration | inform NHS mental health team of concerns about a client. | The NHS mental health team is updated on a client that is deteriorating. |

### Specify Business Use Cases

## 

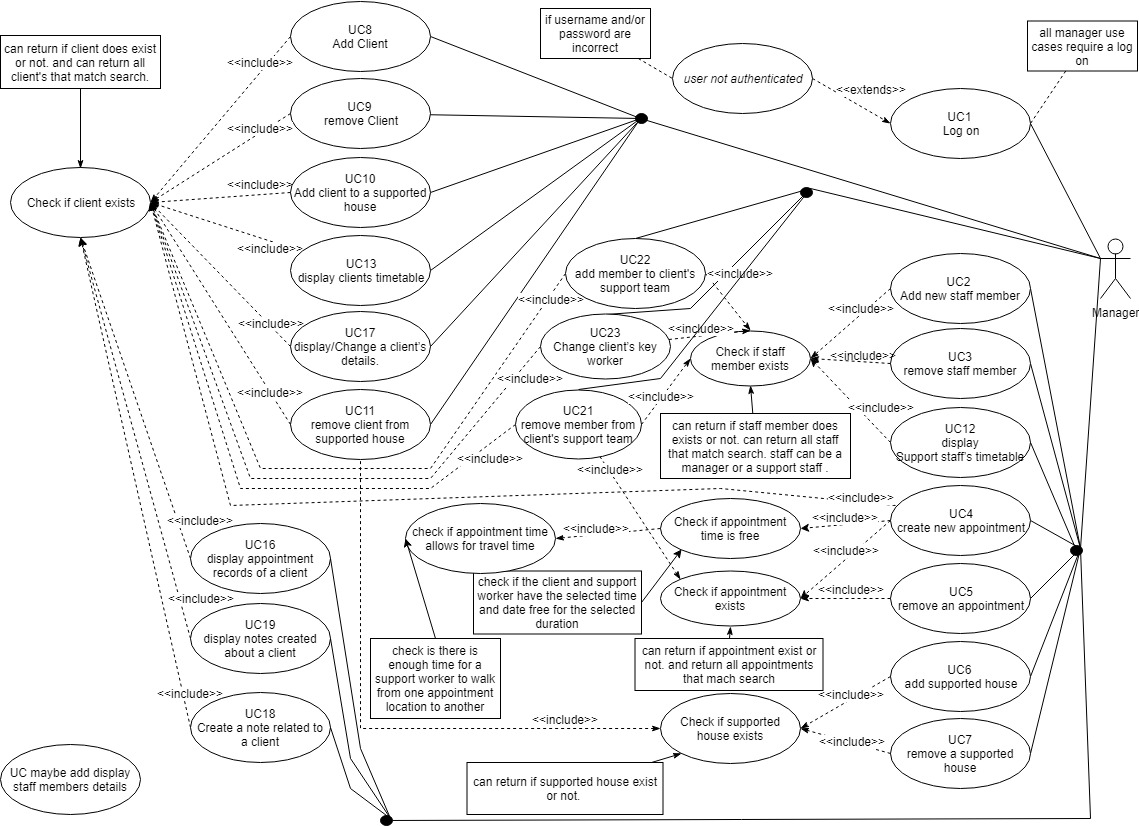
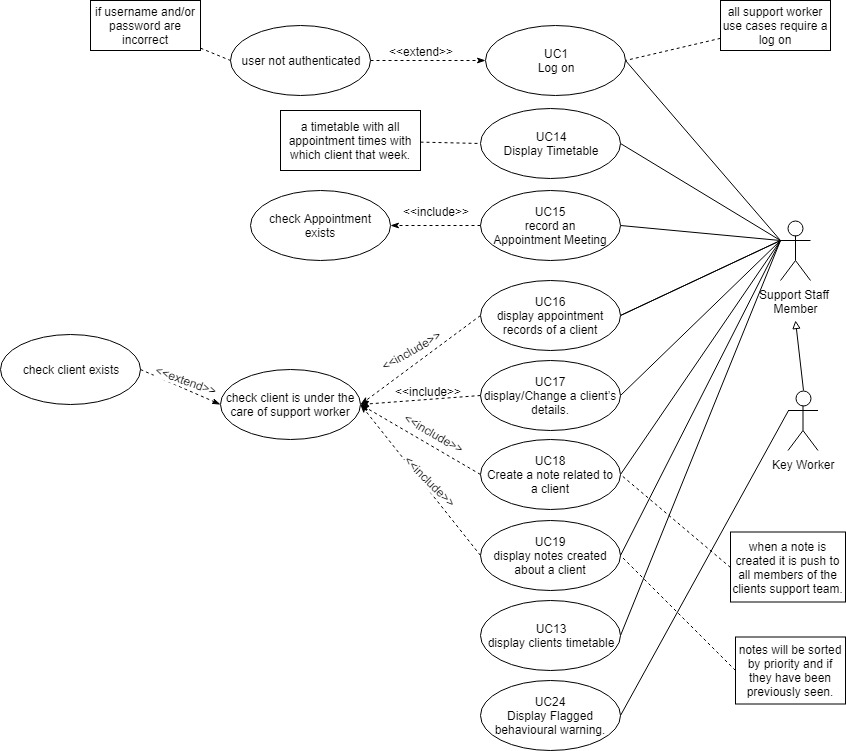
## 

## 

## 

## 

## The Scope of the Product



### Product Use Case Table

|  |  |  |
| --- | --- | --- |
| PUC Number | PUC name | Actor(s) |
| 1 | *log-in* | Manager, Client and support staff member |
| 2 | *Add new staff member* | Manager |
| 3 | *remove staff member* | Manager |
| 4 | *create a new appointment* | Manager |
| 5 | *remove an appointment* | Manager |
| 6 | *add supported house* | Manager |
| 7 | *remove a supported house.* | Manager |
| 8 | *Add Client* | Manager |
| 9 | *remove Client* | Manager |
| 10 | *Add client to a supported house* | Manager |
| 11 | *remove a client from supported house* | Manager |
| 12 | *display Support staff’s timetable.* | Manager |
| 13 | *display client’s timetable.* | Manager and support staff member |
| 14 | *display timetable* | support staff member |
| 15 | *Record an Appointment Meeting.* | support staff member |
| 16 | *display appointment records of a client* | support staff member |
| 17 | *display a client’s details.* | Manager and support staff member |
| 18 | *Create a note related to a client* | support staff member |
| 19 | *display notes created about a client* | support staff member |
| 20 | *display support team information* | Client |
| 21 | *remove a member from the client's support team* | Manager |
| 22 | *add a member to client's support team* | Manager |
| 23 | *Change the client’s key worker* | Manager |
| 24 | *Display Flagged behavioural warning.* | support staff member |

### Individual Product Use Cases

#### **Identifier and Name**: UC1 log-in

**Initiator**: Manager, support staff member or client

**Goal**: the user is identified and logged-in into the system

**Precondition**: None (there are no conditions to be satisfied before this use case can

be carried out.)

**Postcondition**: the user is authenticated, and the dashboard is displayed

for the current user level.

**Assumptions**: the initiator is using a web browser to perform the use case.

**Main Success Scenario**

1. The initiator makes a request to login.
2. The initiator provides their username and password.
3. The system checks the username against the password.
4. The system accepts the username and password and informs the initiator.
5. The initiator is logged-in to the system at the user level (Manager, support staff member or client).

**Extensions**

3.a.1 *user not authenticated.* Inform the user that the username and/or password do not match

#### **Identifier and Name**: UC2 Add new staff member

**Initiator**: Manager

**Goal**: Add a new staff member to the system.

**Precondition**: The Manager is logged-in to the system.

**Postcondition**: A new staff member will be added to the system. with their,

First and last name, date of birth, sex, home address, national Insurance number, position (support staff member or manager) and area of experience. (general mental health, learning disabilities, forensic mental health).

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to create a new staff member.
2. The manager inputs the support staff members first name, last name, date of birth, sex, home address, national Insurance number, position (support staff member or manager) and area of experience. (general mental health, learning disabilities, forensic mental health).
3. The system checks the First name, last name and date of birth number against current staff members.

A3.1 The system checks the national Insurance number against current staff members.

1. The new staff member is recorded, and a username and password are returned.
2. The manager provides the username and password to the staff member.

**Extensions**

3.a.1 *staff member already exists.* Manager is informed that a staff member record exists with the same First name, last name, date of birth, sex and national Insurance number.

#### **Identifier and Name**: UC3 remove staff member

**Initiator**: Manager

**Goal**: remove a member of staff (Manager or support staff member)

from the system.

**Precondition**: A manager account is logged-in. The member of staff to be removed

has no appointments or assign to any clients. A manager cannot

remove themselves.

**Postcondition**: the staff member will be removed from the system.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to remove a staff member.
2. The manager inputs the staff members first name, last name and date of birth

A2.1 The manager inputs the staff members national Insurance number.

1. The system checks the first name, last name and date of birth against current staff members.

A3.1 The system checks the national Insurance number against current

staff members

1. The system displays the first name, last name, date of birth, sex, home address, national Insurance number, position (support staff member or manager) and area of experience. (general mental health, learning disabilities, forensic mental health).

of the staff member that meets the provided information.

1. The manager confirms that is the correct staff member and requests their removal.
2. The system removes that staff member

**Extensions**

3.a.1 *no staff member exists.* the system informs the manager.

4.a.1 *multiple staff members found.* the system displays all matching staff members.

4.b.1 The manager selects the staff member to remove

#### **Identifier and Name**: UC4 create new appointment

**Initiator**: Manager

**Goal**: create an appointment between a support staff member and a client

on a specified day and at a specified time.

**Precondition**: A Manager account is logged-in. There must be an association

between the client and support staff member.

**Postcondition**: an association created between an appointment, support staff

member and a client on a specified day and at a specified time.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to create an appointment between a staff member and a client.
2. The manager inputs the client’s first name, last name and date of birth.
3. The system displays all clients that match the provided first name, last name and date of birth.
4. The manager selects the correct client.
5. The system displays all support staff members with an association to the client.
6. The manager selects the support staff member to create an appointment with.
7. The manager selects the day, time and length of the appointment.
8. The system checks that the client and support staff member are both free on the selected day and time.
9. The system checks that there is time for the support staff member to arrive from previous appointments to the new appointment on time.
10. The system records the new appointment and informs the manager.

**Extensions**

3.a.1 *no client exists.* the system informs the manager.

6.a.1 *no support staff member exists.* the system informs the manager.

8.a.1 *appointment time not available.* The system displays any alterative time slots on the selected day that are available and have travel time.

9.a.1 *not sufficient travel time.*  Inform the manager that travel time is not sufficient for selected time. The system displays any alterative time slots on the selected day that are available and have travel time.

#### **Identifier and Name**: UC5 remove an appointment

**Initiator**: Manager

**Goal**: remove an appointment between a client and support staff member.

**Precondition**: A Manager account is logged-in. the client and staff member exist and

are associated.

**Postcondition**: the appointment between a client and support staff is removed.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to remove an appointment.
2. The manager inputs the client’s first name, last name, date of birth and the day and time of the appointment.

A2.1 The manager inputs the support staff member’s first name, last name,

date of birth and the day and time of the appointment.

1. The system checks for the client’s appointment on the selected day and time.

A3.1 The system checks for a support staff member’s appointment on the selected

day and time.

1. The system displays the day, time and length of the selected appointment.
2. The manager confirms that is the correct appointment and requests its removal.
3. The system removes that appointment from both the client and staff member.

**Extensions**

3.a.1 *no appointment found.* the system informs the manager.

#### **Identifier and Name**: UC6 add supported house.

**Initiator**: Manager

**Goal**: record a new support house to the system.

**Precondition**: A Manager account is logged-in.

**Postcondition**: the new support house is added to the system.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to add a support house.
2. The manager inputs the address of the house, number of rooms, if gender restricted and if for specific conditions (general mental health, learning disabilities, forensic mental health) and a supported house name.
3. The system checks if there are already a supported house with that same address.
4. The system records the new support house with the provided information.
5. The system informs the manager that the new support house was successfully added.

**Extensions**

3.a.1 *supported house already exists.* the system informs the manager.

#### **Identifier and** **Name**: UC7 remove a supported house.

**Initiator**: Manager

**Goal**: remove a support house from the system.

**Precondition**: A Manager account is logged-in.the house has no assigned clients.

**Postcondition**: the record of a supported house is removed.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to remove a supported house.
2. The manager inputs the address of the house or support house name
3. The system checks if the supported house exists.
4. The system displays the support house that was found.
5. The manager checks the house and confirms the supported house removal.

**Extensions**

3.a.1 *supported house does not exist.*  the system informs the manager.

#### **Identifier and Name**: UC8 Add Client.

**Initiator**: Manager

**Goal**: adds a client to the system.

**Precondition**: A Manager account is logged-in.

**Postcondition**: a client record is added to the system.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to add a client to the system.
2. The manager inputs the clients first and last name, NHS number, Date of birth, next of kin and general condition (general mental health, learning disabilities, forensic mental health) and diagnosis if available.
3. The system checks if the client with the same first and last name, NHS number and Date of birth exist.
4. The system records the client with the supplied information. And informs the manager of completion

**Extensions**

3.a.1 *client already exists.*  the system informs the manager.

#### **Identifier and Name**: UC9 remove Client.

**Initiator**: Manager

**Goal**: remove a client from the system.

**Precondition**: A Manager account is logged-in.the client is not associated with any

support staff member supported house or appointment. Except for the support staff member associated as their key worker.

**Postcondition**: a client record is removed from the system.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to remove a client from the system.
2. The manager inputs clients first, last name and date of birth.
3. The system checks for clients with the selected same first, last name and date of birth.
4. The system displays any matching clients
5. The manager selects which client to remove.
6. The system removes the client. and informs the manager.

**Extensions**

3.a.1 no *client exists.*  the system informs the manager.

#### **Identifier and Name**: UC10 Add client to a supported house.

**Initiator**: Manager

**Goal**: add a client to a supported house.

**Precondition**: A Manager account is logged-in**.**

**Postcondition**: a record of a client being assigned to a support house created.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to add a client to a supported house.
2. The manager inputs the client’s first name, last name, date of birth.
3. The system checks that a client exists with the selected same first, last name and date of birth.
4. The system checks that the client is not already associated with a supported house.
5. The manager inputs the supported house name or address.
6. The system checks that the support house exists.
7. The system checks if the selected supported house has any free rooms.
8. The system displays to the user which rooms are free.
9. The manager chooses a free room.
10. The system creates the association between the client and supported house and the room.
11. The system displays a confirmation.

**Extensions**

3.a.1 *no* *client exists.*  the system informs the manager.

4.a.1 *client already in a support house.* The system informs the manager.

4.b.1 the system offers to remove the current supported house and replace it with a newly

Selected support house.

6.a.1 *supported house does not exist.* The system informs the manager.

7.a.1 *No free rooms*. The system informs the manager.

#### **Identifier and Name**: UC11 remove client from supported house.

**Initiator**: Manager

**Goal**: remove a client from a supported house.

**Precondition**: A Manager account is logged-in.

**Postcondition**: the association between a client and supported house is removed.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to remove a client from a supported house.
2. The manager inputs clients first name, last name and date of birth.
3. The system checks that the client exists with the selected same first, last name and date of birth.
4. The system checks which supported house the client is associated with.
5. The system displays the support house the client is associated with to the manager.
6. The manger confirms the removal.
7. The system removes the association records the room as free and confirm to the manager.

**Extensions**

3.a.1 *no* *client exists.*  the system informs the manager.

4.a.1 *client not associated with a supported house.* The system informs the manager.

#### **Identifier and Name**: UC12 display Support staff’s timetable.

**Initiator**: manager

**Goal**: display the timetable for a selected support staff member.

**Precondition**: A Manager account is logged-in.

**Postcondition**: the selected timetable is displayed to the manager.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to receive a support staff member’s timetable.
2. The manager inputs the support staff members first name, last name and date of birth.
3. The system checks that the support staff member exists with the selected same first, last name and date of birth.
4. The system retrieves appointments the selected support staff member is associated with.
5. The system displays the support staff member’s timetable with the day, time, client’s first name, last name and location.
6. The manager selects an appointment from the list to display the appointments details and recorded appointment meetings.
7. The system displays the selected appointment details along with all recorded appointment meetings.

**Extensions**

3.a.1 *no support staff member exists.*  the system informs the manager.

#### **Identifier and Name**: UC13 display client’s timetable.

**Initiator**: manager and support staff member

**Goal**: display the timetable for a selected client.

**Precondition**: the initiator must be logged-in. the support staff member must

be associated with the client (they must be part of their care team)

**Postcondition**: the selected timetable is displayed to the Initiator.

**Assumptions**: The Initiator is using a web browser to perform the use case.

**Main Success Scenario**

1. The Initiator makes a request to receive a client’s timetable.
2. The Initiator inputs the client’s first name, last name and date of birth.
3. The system checks that the client exists with the selected same first name, last name and date of birth.

A3.1 The system checks that the client exists with the selected same first name, last name and date of birth and is associated with the support staff member that is logged-in.

1. The system retrieves appointments the selected client is associated with.
2. The system displays the client’s timetable with the day, time, location and support staff member’s first and last name.

**Extensions**

3.a.1 *no client* *exists.*  the system informs the Initiator.

A3.1.a.1 *Support staff member not associated with client*. the system informs the Initiator

#### **Identifier and Name**: UC14 display timetable.

**Initiator**: support staff member or client

**Goal**: display the timetable for the logged-in support staff member or client.

**Precondition**: The Initiator must be logged-in**.**

**Postcondition**: The Initiator’s timetable is displayed.

**Assumptions**: The Initiator is using a web browser to perform the use case.

**Main Success Scenario**

1. The Initiator makes a request to receive their timetable.
2. The system retrieves appointments the Initiator is associated with.
3. The system displays the support staff member’s timetable with the day, time, client’s first name, last name and location.

A3.1 The system displays the logged-in client’s timetable with the day, time, support staff member’s first name, last name.

1. The logged-in support staff member can select an appointment from the list to display the appointments details and recorded appointment meetings.
2. The system displays the selected appointment details along with all recorded appointment meetings to the logged-in support staff member.

**Extensions**

2.a.1 *no appointments.*  the system informs the support staff member.

#### **Identifier and Name**: UC15 record an Appointment Meeting.

**Initiator**: Support Staff Member

**Goal**: record the date of the appointment meeting the time support started,

finished and what was done for the supported time.

**Precondition**: the support staff member must be logged-in**.** an association must exist

between a client, support staff member

and appointment.

**Postcondition**: a record the date of the appointment meeting the time support

started, finished and what was done for the supported time is created

and an associated with an appointment.

**Assumptions**: The Support Staff Member is using a web browser to perform the use

case.

**Main Success Scenario**

1. The support staff member makes a request to record an appointment meeting.
2. The system displays the support staff member’s timetable with the day, time, client’s first, last name and location.

A2.1 The support staff member selects the day of the appointment.

A2.2 The system displays all appointments for that support staff member on the selected day.

1. The support staff member selects the appointment they wish to make an appointment meeting record for.
2. The support staff member inputs the date of the meeting the time it started, finished and what was done for the supported time.
3. The system records the appointment meeting record and confirms to the support staff member.

**Extensions**

5.a.1 *Appointment meeting already recorded for that specific date and time*. The system inform the Support Staff Member.

#### **Identifier and Name**: UC16 display appointment meeting records of a client.

**Initiator**: Support Staff Member or Manager

**Goal**: display a specific appointment meeting record of a client.

**Precondition**: The Initiator must be logged-in**.** an association must

exist between a client and support staff member (the support staff

member must be part of the client’s support team.).

**Postcondition**: a specific appointment meeting record of a client is displayed.

**Assumptions**: The Initiator is using a web browser to perform the use

case.

**Main Success Scenario**

1. The Initiator makes a request to display a specific appointment meeting record.
2. The system displays all clients the supported staff member is associated with (all clients the support staff member is supporting).

A2.1 The Manager inputs the First name, Last name of the client they wish to see the meeting records of.

1. The Initiator selects the client they want.
2. The system displays all appointments of that client.
3. The Initiator selects the appointment they which to check.
4. The system displays all appointment meeting dates for that appointment.
5. The Initiator selects the appointment meeting they which to display.
6. The system displays that appointment meeting’s start time, finish time and what was done for the supported time.

**Extensions**

A2.1.a.1 *no client* *exists.*  the system informs the manager.

#### **Identifier and Name**: UC17 display/Change a client’s details.

**Initiator**: manager and support staff member

**Goal**: display the selected client’s details, first and last name, their

supported house, date of birth and general mental health.

(general mental health, learning disabilities, forensic mental health) and diagnosis if available.

**Precondition**: The Initiator must be logged-in. the support staff member must

be associated with the client (they must be part of their care team)

**Postcondition**: display the client’s details, first and last name, their supported house,

date of birth, NHS number and general mental health.

(general mental health, learning disabilities, forensic mental health)

and diagnosis if available

**Assumptions**: The Manager or support staff member is using a web browser to

perform the use case.

**Main Success Scenario**

1. The Initiator makes a request to receive a client’s details.
2. The Initiator inputs the client’s first name, last name and date of birth.
3. The system checks that the client exists with the selected same first name, last name and date of birth and if the Initiator is a support staff member there must be an association between the them and the client.
4. The system retrieves the client’s details.
5. The system displays the client’s details, first name, last name, their supported house, specific room, date of birth, NHS number and general mental health. (general mental health, learning disabilities, forensic mental health) and diagnosis if available.
6. The system allows the Manager to change the clients first name, last name, date of birth and general mental health. (general mental health, learning disabilities, forensic mental health) and diagnosis.

A6.1 if the support staff member is associated as the client’s key worker then the system will allow them to change the clients first name, last name, date of birth and general mental health. (general mental health, learning disabilities, forensic mental health) and diagnosis.

**Extensions**

3.a.1 *no client* *exists.*  the system informs the user.

3.b.1 *support staff member not associated*. The system informs the user.

3.c.1 *not authorised to access record*. The system informs the user.

#### **Identifier and Name**: UC18 Create a note related to a client.

**Initiator**: support staff member or Manager.

**Goal**: create a note about a client with a priority to be displayed to other

Members of the clients support team.

**Precondition**: The Initiator must be logged-in. a support staff member must be

associated with the client (they must be part of their care team).

**Postcondition**: A note is created about a client with a priority to be pushed to other

members of the client’s support team

**Assumptions**: The Initiator is using a web browser to perform the use case.

**Main Success Scenario**

1. The Initiator makes a request to create a note about a client.
2. The system retrieves all clients associated with the logged-in support staff member.

A2.1 The Manager inputs the First name, Last name of the client they wish to create a note for.

1. The Initiator selects the client to create a note about.
2. The Initiator inputs the note and selects a priority.
3. The system records the note, associates it with the client and pushes the note to all members of that client’s support team.
4. The system confirms this is the Initiator.

**Extensions**

A2.1.a.1 *no client* *exists.*  the system informs the manager.

#### **Identifier and Name**: UC19 display notes created about a client.

**Initiator**: support staff member or Manager

**Goal**: display notes about a client.

**Precondition**: The Initiator must be logged-in. the support staff member must be

associated with the client (they must be part of their care team)

**Postcondition**: all notes associated with a client are displayed

**Assumptions**: The Initiator is using a web browser to perform the use case.

**Main Success Scenario**

1. The Initiator makes a request display notes for a client.
2. The system retrieves all clients associated with the logged in support staff member.

A2.1 The Manager inputs the First name, Last name of the client they wish to create a note for.

1. The Initiator selects the client to display the notes of.
2. The system displays all note created about the selected client.
3. The Initiator selects how they wish the notes displayed (newest first, newest unread first or newest based on priority).

**Extensions**

A2.1.a.1 *no client* *exists.*  the system informs the manager.

#### **Identifier and Name**: UC20 display support team information.

**Initiator**: Client

**Goal**: display the logged-in clients support team details. The first and last

name of each support team member and their contact information.

**Precondition**: None (there are no conditions to be satisfied before this use case can

be carried out.)

**Postcondition**: display the logged in clients support team details. The first and last

name of each support team member and their contact information.

**Assumptions**: client is using a web browser to perform the use case.

**Main Success Scenario**

1. The client makes a request to receive the details of their support team.
2. The system retrieves the support staff members associated with the client.
3. The system displays each support staff members first and last name their contact information and which is the key worker.

**Extensions**

3.a.1 no *support staff member’s associated*. The system informs the user.

#### **Identifier and Name**: UC21 remove member from client's support team

**Initiator**: Manager

**Goal**: remove a member of a client’s support team.

**Precondition**: The manager must be logged-in. The client is associated with support

staff members.

**Postcondition**: support staff member(s) is removed from the client’s support team.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to remove a member of a client’s support team.
2. The manager inputs the First and last name, date of birth of the client.

A2.1 The manager inputs the First and last name, date of birth and NHS number.

1. The system checks the First and last name and date of birth against current clients.
2. The system displays all support staff members assigned to the client.
3. The manager selects the support staff member to be removed.
4. The system check that the selected support staff member has no appointments with the client.
5. The system removes the association between the client and support staff member.
6. The system informs the manager of completion.
7. The system asks if the manager which to remove more support staff members.

**Extensions**

2.a.1 *multiple clients found.* System displays all match clients.

2.b.1 manager selects the correct client.

3.a.1 *no client* *exists.*  the system informs the manager.

6.a.1 *appointments exist with client.* the system informs the manager.

6.b.1 the system asks if the support staff member should still be removed.

6.c.1 if yes, the system removes the appointments and continues.

#### **Identifier and Name**: UC22 add member to client's support team

**Initiator**: Manager

**Goal**: adda member of a client’s support team.

**Precondition**: Manger must be logged-in.

**Postcondition**: support staff member(s) is added to the client’s support team.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to add a member of a client’s support team.
2. The manager inputs the First and last name, date of birth of the client.
3. The system checks the First and last name and date of birth against current clients.
4. The system displays all eligible support staff members (support staff with the same area of experience. general mental health, learning disabilities or forensic mental health).
5. The manager selects the support staff member to be added.
6. The system creates the association between the client and support staff member.
7. The system informs the manager of completion.
8. The system asks if the manager which to add more support staff members.

**Extensions**

3.a.1 *no client* *exists.*  the system informs the manager.

*4.a.1* *no eligible support staff*. the system informs the manager.

#### Identifier and Name: UC23 Change client’s key worker

**Initiator**: Manager

**Goal**: change the support staff member that is assigned as the key worker

for a client.

**Precondition**: Manger must be logged-in.

**Postcondition**: the association between a client and key worker is added or replaced.

**Assumptions**: The Manager is using a web browser to perform the use case.

**Main Success Scenario**

1. The manager makes a request to Change/add client’s key worker.
2. The manager inputs the First and last name, date of birth of the client.
3. The system checks the First and last name and date of birth against current clients.
4. The system asks if the new key worker is part of the current support team.
5. If the manager say no. The system displays all eligible support staff members (support staff with the same area of experience. general mental health, learning disabilities or forensic mental health).

A5.1 If the manager says yes. The system displays the current support team of the

client.

1. The manager selects the support staff member to be added/changed.
2. The system checks the selected support staff member to become the new key worker is not associated with more than three clients.
3. If the client currently has a support staff member assigned as their key worker, the system will remove it.
4. The system creates the association between the client and selected support staff member and associates them as the key worker.

A9.a.1 The system assigns the selected clients associated support staff member as their key worker.

1. The system informs the manager of completion.

**Extensions**

3.a.1 *no client* *exists.*  the system informs the manager.

*5.a.1 no eligible support staff*. the system informs the manager.

*7*.a.1 *selected support staff member is associated with three clients as their key worker*. The system informs the manager

#### Identifier and Name: UC24 Display Flagged behavioural warning.

**Initiator**: System

**Goal**: inform a client’s key worker of a behavioural flag.

**Precondition**: support staff member must be logged-in. the support staff member

must be associated as the key worker of a client.

**Postcondition**: A warning is displayed on the support staff members device.

**Assumptions**: The support staff member is using a web browser to perform the use

case.

**Main Success Scenario**

1. When an appointment meeting note is created the system will check if the appointment is cancelled.
2. The system will check if the client has cancelled their last three appointments.

A2.1 The will check if the client has cancelled their last three appointments with a specific support staff member.

1. The system will inform the client’s key worker that the client has cancelled their last three appointments.

A3.1 The system will inform the client’s key worker that the client has cancelled their last three appointments with a specific support staff member.

## Functional Requirements

Software Functional requirements

[SFR1: UC1, Step 1 40](#_Toc3372160)

[SFR2: UC1, Step 2 40](#_Toc3372161)

[SFR3: UC1, Step 2 40](#_Toc3372162)

[SFR4: UC1, Step 3 40](#_Toc3372163)

[SFR5: UC1, Step 5 40](#_Toc3372164)

[SFR6: UC1, Step 3.a.1 40](#_Toc3372165)

[SFR7: UC2, Step 2 40](#_Toc3372166)

[SFR8: UC2, Step 2 40](#_Toc3372167)

[SFR9: UC2, Step 2 40](#_Toc3372168)

[SFR10: UC2, Step 2 41](#_Toc3372169)

[SFR111: UC2, Step 2 41](#_Toc3372170)

[SFR12: UC2, Step 2 41](#_Toc3372171)

[SFR13: UC2, Step 3 41](#_Toc3372172)

[SFR14: UC2, Step 4 41](#_Toc3372173)

[SFR15: UC2, Step 4 41](#_Toc3372174)

[SFR16: UC2, Step 4.a.1 41](#_Toc3372175)

[SFR17: UC3, Step 3 42](#_Toc3372176)

[SFR18: UC3, Step 4 42](#_Toc3372177)

[SFR19: UC3, Step 6 42](#_Toc3372178)

[SFR20: UC3, Step 3.a1 42](#_Toc3372179)

[SFR21: UC3, Step 4.a1 42](#_Toc3372180)

[SFR22: UC3, Step 4.b1 42](#_Toc3372181)

[SFR23: UC4, Step 7 42](#_Toc3372182)

[SFR24: UC4, Step 7 42](#_Toc3372183)

[SFR25: UC4, Step 7 43](#_Toc3372184)

[SFR26: UC4, Step 8 43](#_Toc3372185)

[SFR27: UC4, Step 8 43](#_Toc3372186)

[SFR28: UC4, Step 9 43](#_Toc3372187)

[SFR29: UC4, Step 9 43](#_Toc3372188)

[SFR30: UC4, Step 3.a.1 43](#_Toc3372189)

[SFR31: UC4, Step 8.a.1 43](#_Toc3372190)

[SFR32: UC4, Step 9.a.1 44](#_Toc3372191)

[SFR33: UC5, Step 3 44](#_Toc3372192)

[SFR34: UC5, Step 4 44](#_Toc3372193)

[SFR36: UC5, Step 6 44](#_Toc3372194)

[SFR37: UC5, Step 3.a.1 44](#_Toc3372195)

[SFR38: UC6, Step 2 44](#_Toc3372196)

[SFR39: UC6, Step 2 44](#_Toc3372197)

[SFR40: UC6, Step 2 44](#_Toc3372198)

[SFR41: UC6, Step 2 45](#_Toc3372199)

[SFR42: UC6, Step 4 45](#_Toc3372200)

[SFR43: UC6, Step 3.a.1 45](#_Toc3372201)

[SFR44: UC7, Step 2 45](#_Toc3372202)

[SFR45: UC7, Step 3.a.1 45](#_Toc3372203)

[SFR46: UC8, Step 2 45](#_Toc3372204)

[SFR47: UC8, Step 2 45](#_Toc3372205)

[SFR48: UC8, Step 3 45](#_Toc3372206)

[SFR49: UC9, Step 5 45](#_Toc3372207)

[SFR50: UC10, Step 7 46](#_Toc3372208)

[SFR51: UC10, Step 9 46](#_Toc3372209)

[SFR52: UC10, Step 4.a.1 46](#_Toc3372210)

[SFR53: UC10, Step 7.a.1 46](#_Toc3372211)

[SFR54: UC11, Step 7 46](#_Toc3372212)

[SFR55: UC12, Step 4 46](#_Toc3372213)

[SFR56: UC12, Step 5 46](#_Toc3372214)

[SFR57: UC12, Step 7 46](#_Toc3372215)

[SFR58: UC15, Step 4 47](#_Toc3372216)

[SFR59: UC15, Step 4 47](#_Toc3372217)

[SFR60: UC15, Step 4 47](#_Toc3372218)

[SFR61: UC15, Step 5 47](#_Toc3372219)

[SFR62: UC17, Step 3 47](#_Toc3372220)

[SFR63: UC17, Step 3 47](#_Toc3372221)

[SFR64: UC17, Step 5 47](#_Toc3372222)

[SFR65: UC17, Step 3.c.1 48](#_Toc3372223)

[SFR66: UC18, Step 4 48](#_Toc3372224)

[SFR67: UC18, Step 4 48](#_Toc3372225)

[SFR68: UC18, Step 5 48](#_Toc3372226)

[SFR69: UC18, Step 5 48](#_Toc3372227)

[SFR70: UC19, Step 2 48](#_Toc3372228)

[SFR71: UC19, Step 4 48](#_Toc3372229)

[SFR72: UC19, Step 5 48](#_Toc3372230)

[SFR73: UC19, Step 5 49](#_Toc3372231)

[SFR74: UC19, Step 5 49](#_Toc3372232)

[SFR75: UC20, Step 2 49](#_Toc3372233)

[SFR76: UC20, Step 3 49](#_Toc3372234)

[SFR77: UC20, Step 3.a.1 49](#_Toc3372235)

[SFR78: UC21, Step 6 49](#_Toc3372236)

[SFR79: UC21, Step 7 49](#_Toc3372237)

[SFR80: UC21, Step 6.a.1 49](#_Toc3372238)

[SFR81: UC21, Step 6.b.1 50](#_Toc3372239)

[SFR82: UC21, Step 6.c.1 50](#_Toc3372240)

[SFR83: UC21, Step 6.c.1 50](#_Toc3372241)

[SFR84: UC22, Step 4 50](#_Toc3372242)

[SFR85: UC22, Step 4.a.1 50](#_Toc3372243)

[SFR86: UC23, Step 4 50](#_Toc3372244)

[SFR87: UC23, Step 5 50](#_Toc3372245)

[SFR88: UC23, Step 7 51](#_Toc3372246)

[SFR89: UC23, Step 7.a.1 51](#_Toc3372247)

[SFR90: UC23, Step 8 51](#_Toc3372248)

[SFR91: UC23, Step 9 51](#_Toc3372249)

[SFR92: UC23, Step A9.a.1 51](#_Toc3372250)

[SFR93: UC24, Step 1 51](#_Toc3372251)

[SFR94: UC24, Step 2 51](#_Toc3372252)

[SFR95: UC24, Step 2 52](#_Toc3372253)

[SFR96: UC24, Step A2.1 52](#_Toc3372254)

[SFR97: UC24, Step 3 52](#_Toc3372255)

[SFR98: UC24, Step A3.1 52](#_Toc3372256)

## SFR1: UC1, Step 1

**Description:** The System shall allow a Manager, support staff member or client to authenticate themselves to the system.

**Fit Criterion:** The login Page shall be displayed.

## SFR2: UC1, Step 2

**Description:** The System shall accept a valid Username from the initiator.

**Fit Criterion:** The system shall accept a username that is under ten characters long (inclusive).

## SFR3: UC1, Step 2

**Description:** The System shall accept a valid password from the initiator.

**Fit Criterion:** The system shall accept a password that is under ten characters long (inclusive).

## SFR4: UC1, Step 3

**Description:** The System shall check the Username and password provided against the same username and password stored.

**Fit Criterion:** The system shall accept the Initiator as authenticated if they have provided a matching username and password.

## SFR5: UC1, Step 5

**Description:** The System shall accept the login request and informs that initiator.

**Fit Criterion:** The system shall display the logged-in dashboard based on the initiators authorisation (if they are a Manager, support staff member or client).

## SFR6: UC1, Step 3.a.1

**Description:** *user not authenticated*. The System shall reject login request if the username and password obtained from the initiator do not match any stored username and password pair.

**Fit Criterion:** The system shall display the login failed massage to the initiator.

## SFR7: UC2, Step 2

**Description:** The system shall accept a valid First and last name.

**Fit Criterion:** The system shall accept a first and last name that contains letters.

## SFR8: UC2, Step 2

**Description:** The system shall accept a valid date of birth.

**Fit Criterion:** The system shall accept a date of birth in the format DD/MM/YYYY that puts the support staff member or manager over the age of eighteen.

## SFR9: UC2, Step 2

**Description:** The system shall accept a valid sex.

**Fit Criterion:** The system shall accept a sex of male, female, Transgender or Do not identify as female, male, or transgender.

## SFR10: UC2, Step 2

**Description:** The system shall accept a valid home address.

**Fit Criterion:** The system shall accept a home address that exists (is a real address of a real residential property).

## SFR111: UC2, Step 2

**Description:** The system shall accept a valid national Insurance number.

**Fit Criterion:** The system shall accept a valid national Insurance number that exists (is a real national Insurance number).

## SFR12: UC2, Step 2

**Description:** The system shall accept a valid area of experience.

**Fit Criterion:** The system shall accept general mental health, learning disabilities or forensic mental health.

## SFR13: UC2, Step 3

**Description:** The system shall check the provided First name, last name, date of birth, sex and home address against the stored staff members.

**Fit Criterion:** The system shall check if any current staff member (support staff or manager) on record has the same First name, last name, date of birth, sex and home address as the perspective staff member.

## SFR14: UC2, Step 4

**Description:** The system shall record a new staff member with the details First name, last name, date of birth, sex, home address, national Insurance number and area of experience. (general mental health, learning disabilities, forensic mental health).

**Fit Criterion:** The system shall create a new record of a staff member with the provided First name, last name, date of birth, sex, home address, national Insurance number and area of experience. (general mental health, learning disabilities, forensic mental health).

## SFR15: UC2, Step 4

**Description:** The system shall create a new username in the form of first name plus last name and a password for the created staff member.

**Fit Criterion:** The system shall create a new username in the form of first name plus last name and a randomly generated password. To act as authentication for the staff member.

## SFR16: UC2, Step 4.a.1

**Description:** *Support staff member already exists.* The system shall display to the manager that a support staff member with the same first name, last name, date of birth, sex, home address and area of experience exists.

**Fit Criterion:** The system shall display that a staff member with the same First name, last name, date of birth, sex, home address and national Insurance number already exists and will not create a new record.

## SFR17: UC3, Step 3

**Description:** the system shall check first and last name and date of birth against stored staff members.

**Fit Criterion:** the system will match all record that have the same first name, last name and date of birth as they provided.

## SFR18: UC3, Step 4

**Description:** the system shall display the record of a staff member that matches the provided first name, last name and date of birth.

**Fit Criterion:** the system will display the first name, last name, date of birth, sex, home address, national Insurance number, position (support staff member or manager) and area of experience. (general mental health, learning disabilities, forensic mental health).

## SFR19: UC3, Step 6

**Description:** the system accepts a remove request and deletes the staff members record.

**Fit Criterion:** the system shall successfully remove the staff member’s record.

## SFR20: UC3, Step 3.a1

**Description:** *no staff member exists:* the system shall display to the manager that no staff members meets the provided first name, last name and date of birth.

**Fit Criterion:** no staff members exist that first name, last name and date of birth.

## SFR21: UC3, Step 4.a1

**Description:** *multiple staff members found*: the system shall display a list of all members that have match the provided first name, last name and date of birth.

**Fit Criterion:** the system shall correctly match all staff members that match the provided first name, last name and date of birth.

## SFR22: UC3, Step 4.b1

**Description:** the system shall allow the manger to select which staff member they will delete from the list of multiple staff members.

**Fit Criterion:** the system shall accept the input from the manager and display the chosen staff member.

## SFR23: UC4, Step 7

**Description:** the system shall accept a valid day from the manager.

**Fit Criterion:** the system shall accept Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or Sunday from the manager.

## SFR24: UC4, Step 7

**Description:** the system shall accept a valid Time from the manager.

**Fit Criterion:** the system shall accept a time from 8am to 6pm inclusive.

## SFR25: UC4, Step 7

**Description:** the system shall accept a valid length of the appointment.

**Fit Criterion:** the system shall accept a length of the appointment from 30 minutes to 2 hours.

## SFR26: UC4, Step 8

**Description:** the system shall check that the client is free on the selected day and time.

**Fit Criterion:** the system shall correctly display if a client is free.

## SFR27: UC4, Step 8

**Description:** the system shall check that the support staff member is free on the selected day and time.

**Fit Criterion:** the system shall correctly display if a support staff member is free.

## SFR28: UC4, Step 9

**Description:** the system shall check that the support staff member has enough available time to walk from their last apartment to the new appointment.

**Fit Criterion:** the system shall correctly calculate the time needed to get from the support staff members last appointment to the new appointment and if this is possible.

## SFR29: UC4, Step 9

**Description:** the system shall create a new appointment on the selected day and time between the selected client and support staff member.

**Fit Criterion:** the system shall correctly record the new appointment the on the selected day and time between the selected client and support staff member.

## SFR30: UC4, Step 3.a.1

**Description:** *no client exists:*  the system shall inform that manager that no client exists with the provided First name, last name and date of birth.

**Fit Criterion:** the system shall correctly inform that manager that no client exists with the provided First name, last name and date of birth.

## SFR31: UC4, Step 8.a.1

**Description:** the system shall inform the manager of the appointment time not being available and display all available time slots on the selected day for the selected length including travel time.

**Fit Criterion:** the system shall correctly display all available time slots on the selected day for the selected length of appointment including travel time.

## SFR32: UC4, Step 9.a.1

**Description:** the system shall inform the manager of the appointment time not being available and display all available time slots on the selected day for the selected length including travel time.

**Fit Criterion:** the system shall correctly display all available time slots on the selected day for the selected length of appointment including travel time.

## SFR33: UC5, Step 3

**Description:** the system shall check the client/support staff members appointments on the selected day and time match the proved day and time.

**Fit Criterion:** the system shall correctly return if a matching appointment is found.

## SFR34: UC5, Step 4

**Description:** the system shall display the day, time and length of the selected appointment.

**Fit Criterion:** the system shall correctly display the day, time and length of the selected appointment.

## SFR36: UC5, Step 6

**Description:** the system shall remove the appointment between the client and support staff member on the selected day and time.

**Fit Criterion:** the system shall correctly remove the appointment from both the client and support staff member.

## SFR37: UC5, Step 3.a.1

**Description:** *no appointment found*. the system informs the manager that there is no appointment meet the provided details.

**Fit Criterion:** the system shall correctly inform them manager of that appointment not existing.

## SFR38: UC6, Step 2

**Description:** the system shall accept a valid number of rooms.

**Fit Criterion:** the system shall accept value greater than 0 for number of rooms.

## SFR39: UC6, Step 2

**Description:** the system shall accept a valid gender restricted response.

**Fit Criterion:** the system shall accept male only, female only or mixed.

## SFR40: UC6, Step 2

**Description:** the system shall accept a valid response to if for specific condition.

**Fit Criterion:** the system shall accept general mental health, learning disabilities, forensic mental health or mix mental health conditions.

## SFR41: UC6, Step 2

**Description:** the system shall accept a valid supported house name.

**Fit Criterion:** a valid supported house name can contain letters and number but no special characters.

## SFR42: UC6, Step 4

**Description:** the system shall create a new supported house record with the provide details.

**Fit Criterion:** The system shall currently record a new supported house with the provided details.

## SFR43: UC6, Step 3.a.1

**Description:** *supported house already exists.* the system shall inform the user that a supported house with the same provide details exists.

**Fit Criterion:** The system shall currently inform the user that supported house record exists with the provided details.

## SFR44: UC7, Step 2

**Description:** The manager inputs the address of the house or support house name.

**Fit Criterion:** The system shall accept a valid address or supported house name that can contain letters and number but no special characters.

## SFR45: UC7, Step 3.a.1

**Description:** *supported house does not exist.*  the system informs the manager.

**Fit Criterion:** The system shall correctly inform the manager that a support house matching provided address or support house name does not exists.

## SFR46: UC8, Step 2

**Description:** the system shall accept a valid NHS number.

**Fit Criterion:** a valid NHS number is accepted.

## SFR47: UC8, Step 2

**Description:** the system shall accept a valid next of kin.

**Fit Criterion:** a valid next of kin first and last name is accepted.

## SFR48: UC8, Step 3

**Description:** the system shall accept a valid next of kin.

**Fit Criterion:** a valid next of kin first and last name is accepted.

## SFR49: UC9, Step 5

**Description:** the system shall remove the client record from the system.

**Fit Criterion:** The system shall correctly remove the selected client record.

## SFR50: UC10, Step 7

**Description:** the system shall display any free room from the selected house.

**Fit Criterion:** The system shall display all rooms not associated with a client in the selected house.

## SFR51: UC10, Step 9

**Description:** the system will create an association between the supported house, client and the free room.

**Fit Criterion:** The system shall correctly record the selected client being assigned to the selected room on the selected supported house.

## SFR52: UC10, Step 4.a.1

**Description:** *client already in a support house.* The system shall inform the manager.

**Fit Criterion:** The system shall correctly inform the manager that the client is already associated with a room in a supported house.

## SFR53: UC10, Step 7.a.1

**Description:** the system informs the manager if the selected supported house has no free rooms.

**Fit Criterion:** The system shall correctly inform the manager if all rooms in a supported house have an associated client.

## SFR54: UC11, Step 7

**Description:** the system removes the association between a client the supported house and the room the client is associated with.

**Fit Criterion:** the system will correctly remove the client from the supported house’s room and make said room as free.

## SFR55: UC12, Step 4

**Description:** the system shall return all appointments that the selected support staff member is associated with.

**Fit Criterion:** the system will correctly return all the appointments that the supported staff member is associated with.

## SFR56: UC12, Step 5

**Description:** the system shall display for each day and each Client’s their First name, last name, and location of appointment.

**Fit Criterion:** the system will correctly display each day and each Client’s First name, last name, and location of appointment.

## SFR57: UC12, Step 7

**Description:** the system shall display the appointment details for a selected appointment. And all appointment meeting for that appointment.

**Fit Criterion:** the system will correctly display the appointment details and any appointment meetings.

## SFR58: UC15, Step 4

**Description:** the system shall accept a valid time started.

**Fit Criterion:** The system shall accept a valid time in the 24-hour format.

## SFR59: UC15, Step 4

**Description:** the system shall accept a valid time finished.

**Fit Criterion:** The system shall accept a valid time in the 24-hour format.

## SFR60: UC15, Step 4

**Description:** the system shall accept a valid input for the what was done.

**Fit Criterion:** The system shall accept a valid input for ‘the what was done’ category must have less than 400 words.

## SFR61: UC15, Step 5

**Description:** the system shall record the appointment meeting.

**Fit Criterion:** The system shall create an association between an appointment and appointment meeting. And associate the newly created appointment meeting with the client and support staff member.

## SFR62: UC17, Step 3

**Description:** the system shall check that a client exists with the selected first name, last name and date of birth.

**Fit Criterion:** The system shall correctly check if a client exists with the same provided details.

## SFR63: UC17, Step 3

**Description:** the system shall check if an association exists between the selected client and the logged-in support staff member.

**Fit Criterion:** The system shall correctly check if a client exists with the same provided details.

## SFR64: UC17, Step 5

**Description:** the system shall display the selected client’s first name, last name, their supported house, specific room, date of birth and general mental health. (general mental health, learning disabilities, forensic mental health) and diagnosis if available.

**Fit Criterion:** The system shall display the correct client’s first name, last name, their supported house, specific room, date of birth and general mental health. (general mental health, learning disabilities, forensic mental health) and diagnosis if available to the initiator.

## SFR65: UC17, Step 3.c.1

**Description:** *not authorised to access record*. The logged-in support staff member is not a member of the selected clients support team. The system informs the user.

**Fit Criterion:** The system shall correctly inform the user that there is no association between the selected client and logged-in support staff member.

## SFR66: UC18, Step 4

**Description:** the system shall accept a valid note.

**Fit Criterion:** The system shall accept a valid note of more than 1 word and less than 400 inclusive.

## SFR67: UC18, Step 4

**Description:** the system shall accept a valid priority.

**Fit Criterion:** The system shall accept a valid priority of 1 to 5. (1 being minor and 5 being important)

## SFR68: UC18, Step 5

**Description:** the system shall record the note and associate it with the client.

**Fit Criterion:** The system shall create an associated between the note, client and support staff member.

## SFR69: UC18, Step 5

**Description:** the system shall push the note to all members of that client’s support team.

**Fit Criterion:** The system shall display the note on each member of the client’s support team’s dashboard.

## SFR70: UC19, Step 2

**Description:** The system shall display all clients associated with the logged-in support staff member.

**Fit Criterion:** The system shall correctly display only the clients associated with the logged-in support staff member. No other client will be displayed.

## SFR71: UC19, Step 4

**Description:** The system shall display all notes about the specified client.

**Fit Criterion:** The system shall correctly display all notes created about the specified client.

## SFR72: UC19, Step 5

**Description:** The system shall arrange the notes by the date the message was created.

**Fit Criterion:** The system shall correctly arrange the notes by the date the message was created.

## SFR73: UC19, Step 5

**Description:** The system shall arrange the notes by New unread messages.

**Fit Criterion:** The system shall correctly arrange the notes by New unread messages.

## SFR74: UC19, Step 5

**Description:** The system shall arrange the notes by priority.

**Fit Criterion:** The system shall correctly arrange the notes by priority.

## SFR75: UC20, Step 2

**Description:** the system shall return all appointments that the logged-in client is associated with.

**Fit Criterion:** the system will correctly return all the appointments that the logged-in client is associated with.

## SFR76: UC20, Step 3

**Description:** the system shall display for each day and each support staff member’s First name, last name, and location of appointment.

**Fit Criterion:** the system will correctly display all associated appointments in the specified format.

## SFR77: UC20, Step 3.a.1

**Description:** the system shall inform the client that there are not associated appointments to them.

**Fit Criterion:** the system will correctly inform the client that they have no appointments recorded.

## SFR78: UC21, Step 6

**Description:** the system shall check if the support staff member has any appointments with the selected client.

**Fit Criterion:** the system will correctly return if the selected support staff member has any associations between itself, an appointment and client.

## SFR79: UC21, Step 7

**Description:** the system removes the selected support staff member from the selected clients support team.

**Fit Criterion:** the system shall correctly remove the association between the support staff member and the selected client.

## SFR80: UC21, Step 6.a.1

**Description:** the system shall inform the manager that an appointment exists between the selected support staff member and the selected client.

**Fit Criterion:** the system will correctly inform manager that their exists an association between a client, support staff member and appointment.

## SFR81: UC21, Step 6.b.1

**Description:** the system shall ask if the support staff member should still be removed.

**Fit Criterion:** the system will accept yes or no input from the Manager.

## SFR82: UC21, Step 6.c.1

**Description:** the system shall remove any appointment between the selected support staff member and client.

**Fit Criterion:** the system shall remove any associations between the selected support staff member, the selected client and appointments.

## SFR83: UC21, Step 6.c.1

**Description:** the system shall remove the selected support staff member from the client’s support team.

**Fit Criterion:** the system shall remove the association between the support staff member and the selected client.

## SFR84: UC22, Step 4

**Description:** The system displays all eligible support staff members (support staff with the same area of experience. general mental health, learning disabilities or forensic mental health).

**Fit Criterion:** the system shall correctly display eligible support staff members that match the client’s mental health category (general mental health, learning disabilities or forensic mental health).

## SFR85: UC22, Step 4.a.1

**Description:** *no eligible support staff*. the system informs the managerthat there are no eligible support staff member that match the client’s mental health category.

**Fit Criterion:** the system shall correctly inform the managerthat there are no eligible support staff member that match the client’s mental health category.

## SFR86: UC23, Step 4

**Description:** the system shall ask if the new key worker is part of the current support team.

**Fit Criterion:** The system shall accept ‘yes’ or ‘no’.

## SFR87: UC23, Step 5

**Description:** the system shall display all eligible support staff members. Based on the client’s mental health area. (general mental health, learning disabilities or forensic mental health).

**Fit Criterion:** the system will only display support staff members that match the area of experience (general mental health, learning disabilities or forensic mental health).

## SFR88: UC23, Step 7

**Description:** The system shall check if the selected support staff member has more than three clients they act as key workers for.

**Fit Criterion:** the system shall correctly check if the support staff member is associated as a key worker with three clients.

## SFR89: UC23, Step 7.a.1

**Description:** *selected support staff member is associated with three clients as their key worker*. The system informs the user that the selected support staff ember is not eligible to become a keyworker.

**Fit Criterion:** the system shall inform the user that the selected support staff member is not eligible and allow them to select another support staff member.

## SFR90: UC23, Step 8

**Description:** the system shall remove the assigned keyworker of the selected client.

**Fit Criterion:** the system will correctly remove the key worker association between a client and support staff member.

## SFR91: UC23, Step 9

**Description:** The system assigns a new support staff member and then assigns them as the key worker.

**Fit Criterion:** the system will correctly create a new association between a client and support staff member and then associate them as the client’s key worker.

## SFR92: UC23, Step A9.a.1

**Description:** The system assigns the selected support staff member who is currently part of the selected clients support team as their new key worker.

**Fit Criterion:** the system shall associate the selected support staff member as the key worker for the selected client.

## SFR93: UC24, Step 1

**Description:** when a support staff member records a new appointment meeting record. The system will check if the appointment was declined.

**Fit Criterion:** the system shall check of the appointment meeting is recorded as declined.

## SFR94: UC24, Step 2

**Description:** when a new appointment meeting is recorded as declined the system will check the last two appointment meeting records.

**Fit Criterion:** the system shall access the last two appointment meeting records for the client associated with the new appointment meeting record.

## SFR95: UC24, Step 2

**Description:** the system will check of the last two appointment meetings where also declined.

**Fit Criterion:** the system shall check if the last two appointments scheduled of the client are recorded as declined.

## SFR96: UC24, Step A2.1

**Description:** when a new appointment meeting is recorded as declined the system will check the last two appointment meeting records for each associated support staff member.

**Fit Criterion:** the system shall access the last two appointment meeting records for each support staff members associated with the client associated to the new appointment meeting record.

## SFR97: UC24, Step 3

**Description:** The system will inform the client’s key worker that the client has cancelled their last three appointments.

**Fit Criterion:** The system shall display a warning to the clients associated key worker that they have not accepted with last three appointments.

## SFR98: UC24, Step A3.1

**Description:** The system will inform the client’s key worker that the client has cancelled their last three appointments with a specific support staff member.

**Fit Criterion:** The system shall display a warning to the clients associated key worker that they have not accepted with last three appointments with a specific support staff member. The name of that support staff member will be displayed.

# Non-functional Requirements

## Look and feel requirements

#### LF1: The system shall conform to the charity’s already existing website.

Fit Criterion: The system shall be designed with a blue and white forms. A light grey horizontal menu bar. The charity logo in the upper left corner.

Applies: All use cases

#### LF2: the system shall have uncluttered forms on a mobile device.

Fit Criterion: the system shall be readable, have a text size of 12+ and any input from will be displayed vertically and on its own line.

Applies: All use cases

## Usability and humanity requirements.

#### U1a: The system shall be easy for a manager to use.

Fit Criterion: Managers shall be able to learn the system in 2 hours. All trained manager shall be able to create/remove a new client, manager, supported house or Support staff member in 5 minutes assuming that the information used is correct.

Managers shall be able setup or alter a client’s support team in 20 minutes, assuming that the information used is correct.

Managers shall be able to add/remove client appointments within 10 minutes assuming that the information used is correct.

Applies: UC1, UC2, UC3, UC12, UC4, UC5, UC6, UC7, UC22, UC23, UC21, UC8, UC9, UC10, UC13, UC17, UC11, UC16, UC19, UC18

#### U1b: The system shall be easy for a support staff member to use.

Fit Criterion: Support staff members shall be able to learn the system in 1 hour. All trained support staff members shall be able to, record an appointment meeting in 1 minute, Display client details in 30 seconds, create a client note in 1 minute, display notes relating to a client in 30 seconds.

Applies: UC1, UC14, UC15, UC16, UC17, UC18, UC19, UC13, UC24.

#### U1c: The system shall be easy for a client to use.

Fit Criterion: clients will be able to use the system with 10 minutes of training. Client shall be able to, display their timetable in 30 seconds, display a specific support staff member contact detail’s in 1 minute.

Applies: UC1, UC14, UC20.

#### U2: The system shall conform to the Equality Act 2010.

Fit Criterion: The system shall meet level AA of the Web Content Accessibility Guidelines (WCAG 2.1) as a minimum.

Applies: All use cases.

## Performance requirements.

#### P1: The system shall be able to handle a range of concurrent users.

Fit criterion: The system shall be able to handle up to 100 concurrent users.

Applies: all use cases

#### P2: The system shall be able to handle a range of clients.

Fit criterion: The system shall be able to handle up to 100 clients and their appointments.

Applies: all use cases

#### P3: The system shall respond to user inputs within 2 seconds.

Fit criterion: The system shall be able to respond to more than 50 per cent of users input in under 2 seconds. Assuming no controllable connectivity problems.

Applies: UC1, UC17, UC20, UC8, UC9, UC10, UC11, UC7, UC6, U3, UC2

#### P4: The system shall respond to complex user inputs within 5 seconds.

Fit criterion: The system shall be able to respond to more than 50 per cent of complex users input in under 5 seconds. Assuming no controllable connectivity problems.

Applies: UC13, UC21, UC23, UC23, UC16, UC19, UC18, UC14, UC12, UC15, UC16, UC13, CU4, UC5, UC24.

#### P5: The system shall have high availability.

Fit criterion: The system be availability of 99.9 per cent of the time for 24 hours a day, 7 days a week, with a period of unavailability lasting 20 minutes at most.

Applies: all use cases.

## Operational and environmental requirements.

#### O1: The system shall work on the last four releases of the five most popular browsers.

Fit criterion: the system shall function correctly on the last four releases of the five most popular browsers. Assuming that the device running the browser meets the minimum specification.

Applies: All use cases.

## Maintainability and support requirements.

#### M1: the system shall operate across a range of mobile phones.

Fit criterion: the system shall work on both android and IOS assuming the mobile phone meets the minimum specification and has a browser.

Applies: All use cases.

## Security requirements.

#### S1: only managers shall be able to access manger operations.

Fit criterion: no more than one break per year shall occur.

Applies: UC2, UC3, UC12, UC4, UC5, UC6, UC7, UC22, UC23, UC21, UC8, UC9, UC10, UC11.

#### S2: only support staff members or managers shall be able to access support staff member operations.

Fit criterion: no more than one break per year shall occur.

Applies: UC15, UC16, UC17, UC18, UC19, UC13, UC24.

#### S3: only support staff members, managers or clients shall be able to access client operations.

Fit criterion: no more than one break per year shall occur.

Applies: UC1, UC14, UC20.

#### S4: The system shall prevent incorrect data from being introduced.

Fit criterion: the system shall reject information in the specified format.

Applies: All use cases

#### S5: The system shall reveal private information only in compliance with the organization’s information policy.

Fit criterion: the system shall only show sensitive data to staff involved with that’s clients care.

Applies: All use cases

#### S6: The system shall protect private information in accordance with the relevant privacy laws.

Fit criterion: the system shall protect stored private data from malicious use.

Applies: All use cases

## Cultural requirements.

None.

## Legal requirements

#### L1: the system shall operate in accordance with UK law.

Fit criterion: the system shall be able to pass an audit by a legal compliance company.

Applies: All use cases

#### L2: The system shall be implemented so as to comply with the Data Protection Act.

Fit criterion: the system shall handle personal and sensitive data appropriately.

Applies: All use cases

# Project issues

## Open Issues

An investigation into whether a solution can be implemented around connectivity. Some of the support houses have bad reception. As a website will be used for the front-end it will be necessary to research ways of allowing an offline website to process and/or store information than is then uploaded when a connection is established.

## Off-the-shelf Solutions

Mobizio is a management system in use by several supported living charities. The main drawback of this system is a monthly service charge. Looking at the functionality of this system could help with the development of our software system.

## New problems

Introducing a scheduling system will affect the work of all support staff and managers. the time it takes for the staff to be training and start using the system will depend on its overall acceptation by staff and that is successfully replaces the paper system.

## Tasks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **Task Name** | **Duration** | **start** | **finish** |
| 1 | read TM470 documentation | 1 week | 02-Feb | 09-Feb |
| 2 | **Begin requirements elicitation from support charity.** | 8 weeks | 10-Feb | 31-Mar |
| 3 | *Create requirements specification in line with the Volere template* |  |  |  |
| 4 | *Define Scope and priorities of functionality* |  |  |  |
| 5 | *get feedback from staff and refine.* |  |  |  |
| 6 | complete TMA01 | 1 week | 25-Feb | 04-Mar |
| 7 | send TMA01 | 0 days | 04-Mar | 04-Mar |
| 8 | **begin iteration for basic object representations and appointments recording.** | 11 days | 01-Apr | 12-Apr |
| 9 | *begin analysis and design e.g. system operations, behaviour and structural models* | 4 Days |  |  |
| 10 | *implement iteration* | 4 Days |  |  |
| 11 | *debug and change as needed.* | 3 Days |  |  |
| 12 | complete TMA02 | 1 week | 09-Apr | 16-Apr |
| 13 | send TMA02 | 0 days | 16-Apr | 16-Apr |
| 14 | **begin iteration for note system.** | 5 days | 12-Apr | 17-Apr |
| 15 | *begin analysis and design e.g. system operations, behaviour and structural models* | 2 days |  |  |
| 16 | *implement iteration* | 2 days |  |  |
| 17 | *debug and change as needed.* | 1 days |  |  |
| 18 | research serialization in java and database design | 4 days | 18-Apr | 22-Apr |
| 19 | implement a database to store java objects. | 3 days | 23-Apr | 26-Apr |
| 20 | implement serialization to current java classes and link to database. | 3 days | 27-Apr | 30-Apr |
| 21 | research HTML, CSS, JavaScript, restful and UI design | 2 weeks | 01-May | 15-May |
| 22 | research GDPR compliance, and web accessibility. | 3 days | 16-May | 19-May |
| 23 | create website wireframe and get feedback from staff on design | 3 days | 20-May | 23-May |
| 24 | **begin iteration for website design and restful links to Java back-end.** | 3 weeks | 24-May | 14-Jun |
| 25 | *begin analysis and design e.g. system operations, behaviour and structural models.* | 9 days |  |  |
| 26 | *implement iteration* | 8 days |  |  |
| 27 | *debug and change as needed.* | 4 days |  |  |
| 28 | **begin iteration for admin dashboard.** | 1 week | 15-Jun | 22-Jun |
| 29 | *begin analysis and design e.g. system operations, behaviour and structural models* | 4 days |  |  |
| 30 | *implement iteration* | 2 days |  |  |
| 31 | *debug and change as needed.* | 1 days |  |  |
| 32 | Holiday | 1 week | 24-Jun | 28-Jun |
| 33 | Complete TMA03 | 1 week | 29-Jun | 06-Jul |
| 34 | send TMA03 | 0 days | 09-Jul | 09-Jul |
| 35 | **begin iteration for timetable system.** | 1 week | 10-Jul | 17-Jul |
| 36 | *begin analysis and design e.g. system operations, behaviour and structural models* | 4 days |  |  |
| 37 | *implement iteration* | 2 days |  |  |
| 38 | *debug and change as needed.* | 1 days |  |  |
| 39 | **begin iteration for behavioural flagging and advisory system.** | 1 week | 18-Jul | 25-Jul |
| 40 | *begin analysis and design e.g. system operations, behaviour and structural models* | 3 days |  |  |
| 41 | *implement iteration* | 2 days |  |  |
| 42 | *debug and change as needed.* | 2 days |  |  |
| 43 | research graph route planning algorithm vs integrating google maps | 1 week | 26-Jul | 02-Aug |
| 44 | **begin iteration of travel time system** | 1 weeks | 03-Aug | 10-Aug |
| 46 | *begin analysis and design e.g. system operations, behaviour and structural models* | 2 Days |  |  |
| 47 | *implement iteration* | 3 Days |  |  |
| 48 | *debug and change as needed.* | 2 Days |  |  |
| 49 | get feedback from other staff on system make any needed changes and finalize the system. | 1 weeks | 11-Aug | 18-Aug |
| 50 | complete EMA | 3 weeks 6 days | 19-Aug | 15-Sep |
| 51 | send EMA | 0 days | 15-Sep | 15-Sep |

### Planning of the Development Phases

1. iteration for basic object representations and appointments

in this iteration the representation of supported house, manager, support staff members and clients will be created with the information they must encapsulate. Appointment will also be created with the required information for appointment recording.

Functional requirements included

Use case: UC2, UC3, UC4, UC5, UC6, UC7, UC8, UC9, UC10, UC11.

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, L2

1. iteration for website design and restful links to Java back-end.

in this iteration the website UI and database w will be designed and implemented. The website UI will be linked to the java back-end and database through Spring and restful architecture.

Use case: UC1, UC17, UC19, UC20.

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, L1, L2, O1, P4, P3, P2, P1, U1a, U1b, U1c, U2.

1. iteration for note system.

in this iteration the representation of a note will be designed. And the system necessary to send that note to support staff members with the correct permissions.

Use case: UC18, UC19.

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, L1, L2.

1. iteration for admin dashboard.

in this iteration functionality for administration of the system will be implemented. Manger use cases.

Use case: UC21, UC22, UC23.

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, U1a, U1b, U1c, U2.

1. iteration for timetable system.

in this iteration the timetable system will be implemented. Displaying and altering appointments for support staff and clients.

Use case: UC12, UC13, UC14, UC15, UC16.

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, U1a, U1b, U1c, U2.

1. iteration for behavioural flagging and advisory system.

in this iteration automated flagged will be designed and implemented.

Use case: UC24, UC22, UC23, UC4

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, L1, L2, O1, P4, P3, P2, P1, U1b, U2.

1. iteration of travel time system

in this iteration travel time between appointments will be implemented into the appointment system.

Use case: UC4

Non-functional requirements included

Use case: S1, S2, S3, S4, S5, S6, L1, L2, O1, P4, P3, P2, P1, U1a, U1b, U1c, U2.

## User documentation and training

User manuals will need to be created for the manager and staff members in the use of the system and their responsibility in is effectiveness.

## Waiting room

Expansion to maintaining and cleaning of supported houses.

Introduce housing staff and maintenance staff.

Expansion to medication control for supported clients.

Expansion to the scheduling of pool vehicles.

Expansion allowing the manager to create custom behavioural and advisory warnings.

a ‘created’ field to the note class

# appendix B

UC3 remove staff member

**Context** Charity::removeStaffMember(sm : Staff))

**pre:**

* - The Manager is logged-in to the system.
* - The SupportStaffMember is not linked to any appointment objects.
* - The SupportStaffMember is not linked to any Client objects.
* The Manager is not the logged-in Managers account.

**Post:**

* - The SupportStaffMember or Manager link with its associated Address is removed.
* **-** The SupportStaffMember or Manager object is removed.
* - a conformation String will be returned.
* - **Otherwise**, a String indicating failure will have been returned.

**Context** Charity::removeStaffMember(sm : Staff)

**pre:**

* - The Manager is logged-in to the system.
* - The SupportStaffMember is not linked to any appointment objects.
* - The SupportStaffMember is not linked to any Client objects.
* The Manager is not the logged-in Managers account.

**Post:**

* **-** The SupportStaffMember or Manager object is removed from the collection.
* - a conformation String will be returned.
* - **Otherwise**, a String indicating failure will have been returned.

#### 

