Group Project - Introduction and Project Setup

Liam Miller 793611 & James Watson 789963 & Alex Gregory 793016 & Joshua Fletcher 792259

Milestone One

Supervisor: Tom Owen

February 2018

Contents

1

Project Overview

The following is a brief overview of the work we are to undertake, who we are undertaking it for, and the purpose behind the work.

1.1 Introduction

1.1.1 The Client

The project is being done on behalf of the NHS health board for various different hospitals including Morriston, Singleton, Neath Port Talbot, and Princess of Wales. The full name of the service is the Abertawe Bro Morgannwg University Health Board. They provide health and social services to a wide area including the city of Swansea and the surrounding area. In particular, we are working under Anne-Marie Hutchingson and she is our primary contact with the health board.

They are working in conjunction with Swansea University in order to develop a selection of health tools. Our primary contact with Swansea University is Dr Tom Owen, who runs the masters year engineering projects.

1.1.2 Scenario

The purpose of the project is to design a mobile application that would assist people with tendon injuries. This will assist them in performing a staged recovery over a period of time using the application as a guide to give instructions. With the health service being stretched thin it is difficult for health professionals to train people to perform the required recovery exercises. Additionally, this project is also being used as an example of where technology can help the NHS perform it's functions with less resources while maintaining patient quality.

The intent of this document is to outline the formal requirements, specifications, and scope of the project to provide a mobile application for physiotherapy instruction. This document will outline methodologies, potential risks, and their mitigations as well as providing a timetable for release of the project. This document will also go on to further detail justifications for the choices made, in brief, in order to provide future context to developers and project managers working on the project in future.

1.2 Project Background

At present the current system does not use digital technology to directly aide the patients, the majority of the technology is only operated by the health professionals. At the start of the physiotherapy sessions a patient will be given several examples of exercises to be done between physiotherapy sessions. These will be demonstrated several times and practised with the physiotherapist so that the patient can be sure they are completing the exercises correctly with a trained professional.

After the first set of sessions, the next session is generally scheduled for 3-6 months later. Between these sessions, the patient is expected to perform the exercises in order to aid their recovery process. When they return for more physiotherapy, the doctor will modify their routine to further their recovery process. This will repeat until the injured person is back to full health.

Occasionally, in order to give patients further aide in the recovery processes, local community classes or sessions will be given on the exercises. These take place outside of the normal physiotherapy sessions. These sessions are designed to help patients refresh their understanding of the exercises, as well as to head off any problems that would crop up due to the infrequency of physiotherapy sessions.

1.2.1 Areas of Concern

There are several issues with the current system that have prompted the creation of this project. Our Contact, Anne-Marie Hutchingson has discussed several problems with the current system with us. We have listed some of these below.

- It is hard to track progress of individuals. This means that often they are reliant on self reporting. This causes issues when a health professional attempts to create a new treatment plan during the most recent checkup. An application would help fix this by providing an impartial timeline of how well a patient was doing. This admittedly does cause issues in that it's still self reporting; however, given that records can not be altered, this makes the records more accurate. It also avoids any memory issues that the patient may have.

- Patients heal at different rates. At the minute everyone gets the same treatment plan, but there's no ability for treatment plans to reflect individual recovery rates without another therapy session. An application would avoid this by making it possible for there to be a staged progression based on how well they are doing the exercises.

- Often patients stop doing the exercises properly due to the fact that they are very repetitive; however, with only limited time in each physiotherapy session, only a certain selection of exercises can be shown. An application would allow us to upload a selection of different exercises that could be rotated around in order to relieve the monotony of the recovery process.

- Following on from the previous point, another issue that can occur is that patients can fail to perform the exercises correctly. This can also cause issues during the recovery process as the exercises must be performed correctly in order fully heal everything. An application can help here by using the videos to demonstrate the correct procedures the user should follow.

- Additionally, it was found that patients also failed to perform the exercises due to forgetting about them, or putting them off to do in the later. This would cause issues with certain kinds of professionals, such as office workers, as they would not perform the exercises and thus waste time. An application would help resolve this by providing helpful reminders and notices for a user to do their exercises.

- Another major issue that occurs is that a lot of time is often wasted on just catching up on progress. This often means that time is being wasted to ascertain the current state of recovery. An application can help with this by providing a quick checklist of current stats to help speed the conversation along. The result of this is that less time is wasted.

1.3 Project Motivation

At present, the NHS is undergoing a severe crisis. With ageing populations, and no significant funding increases likely to come soon. Considerable pressure is being applied to the health service at the minute [14]. This is especially problematic within Wales, due to the particular pressures that exist in Wales [20]. Such as having on average an older and sicker population, it has proven especially taxing to the health service in the country [2].

One of the negative side effects of this situation is that waiting times have increased inside Wales for many different kinds of operations and procedures [22] meaning that many people are suffering because they have not seen anyone about their health problems. This is also true for physiotherapy sessions and professionals. Patients are often having to wait long periods between sessions. This often causes issues that can prolong treatment and mean more time wasted.

Due to this, many different health boards have been looking into methods to reduce costs or integrate services better in order to realise efficiency savings. The integration policy has been official government policy (both for the UK and Welsh Governments) since 2015 [10]. As well as looking for savings, there have also been initiatives to allow patients to take care of themselves where possible to ease strain on the health service, and allow for a more efficient recovery.

The project idea is sourced from this collaborative and integrated approach to helping the NHS. Swansea University is heavily involved in the local health board, being very close to the training hospital both in terms of geographical and co-operational ties. As a part of these close ties, Dr Tom Owen offered the services of the Masters of Engineering students in order to help relieve the strain and otherwise modernise the NHS services of the local health board.

Our main health board service contact, Anne-Marie Hutchingson, presented this project as one way we could help to improve the health service and make peoples lives easier.

This project was also partially inspired by the Duolingo application, this application is designed to instruct users in learning another language [7]. This service uses reminders, progress tracking and other tools in order to keep people learning the language they have chosen, as well as to retain the information they have already learnt.

One of our main aims is to make the project act in a similar fashion as to Duolingo. An example of how Duolingo looks is shown in figure 1.1.

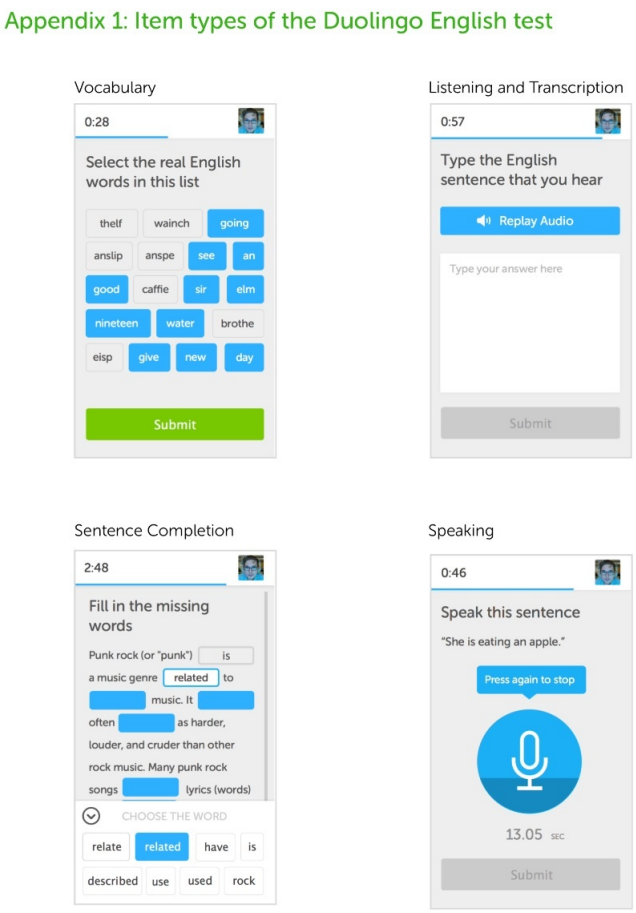

Figure 1.1: Duolingo example, showcasing how Duolingo teaches users about their chosen language.

1.4 Aims and Objectives

The following are our aims and objectives, we intend to accomplish these aims by the end of the project time scale, such that all stakeholders are happy with the progress made.

- To have a system that will work on multiple mobile devices.
  - We intend to use appropriate software to ensure that our system is cross platform.
  - We intend to test the system across multiple devices to ensure quality.
- To track, record, and monitor the progress of a patient over the course of their treatment.
  - The system will be capable of storing, saving, and deleting data via a database instance. This will allow the user to track progress of their exercises.
  - To introduce some form of scoring metric such that we can accurately represent the progress of a patient.

- To allow the physiotherapist to quickly and easily understand where a patient is in terms of their recovery.
  - The system will be capable of performing a full summary of progress, via a specific page or command, such that the patient or medical professional can give accurate and helpful medical advice.
- To remind the patient to perform their exercises on a regular basis.
  - The system will use an inbuilt alert and notification service to remind the users to perform their exercises.
- To prevent patients from becoming bored or tired with their current treatment regime and adapt to match the progress of patients.
  - This will be accomplished by allowing a certain amount of randomisation within the exercises being asked to be performed. This will aim to prevent boredom by removing monotony.
  - This will integrate into the scoring metric and use said metric to enable us to allow slower or quicker progression through the various stages of treatment.

1.5 User Flow

Anne-Marie has given us a brief overview of her ideal flow. The following is how it would work.

The patient would log into the app once prompted by a notification. Once present the patient would perform the given exercises until completion, or until they were unable to continue. The patient would then enter in how well they did, and how many of the exercises they were able to complete. These exercise reports would be used to generate a score.

Over a period of time, to be determined, if their score was good enough they would be allowed to move onto another stage of exercises. If their score was not good enough, then they would have to wait until has improved enough to move on.

During meetings with the physiotherapist, the patient would be able to demonstrate their progress by showing it in a form that the physiotherapist would be able to read. This would allow quicker consultations and better results.

1.6 Application Planning

A series of mock-up designs were created in order to communicate our vision for how the basic design of the application should look. The designs are presented below in figure 1.2. This has allowed us to agree with our client on the high level user experience of the application. Within these designs three of the applications pages were shown: the main exercise display page, a list of exercise progress within that days workout, and a statistics page.

Figure 1.2: A mock-up of the application.

2

Methodology

This section shall address the development methodologies, tools, and standards to be used within this project, alongside justification for each choice made.

2.1 Requirements of the Methodology

The software methodology is considered the backbone of any software development project, it is these methodologies which dictate how the project should be designed, how the implementation processes should take place, how documentation is to be produced, and how and when the software testing phases take place. As this is a group project, it is also important to ensure that the methodology is also suitable for the team, allowing them to work together in an efficient manner.

To ensure that we select a suitable methodology we first need to state the characteristics that we require for a successful methodology.

2.1.1 Requirements of the methodology from a project perspective.

The methodology which is chosen should allow the project to be broken down into multiple smaller components, and then merged together. This is essential within this project as the project shall require both a client interface and an API, these are most likely to be two separate projects, thus rather than needing to completely develop one side and then the other, the ability to break down the project would mean that the two sides could be worked upon simultaneously. Additionally, by being able to break down the project, it becomes more manageable and maintainable, with code more likely being placed within sensible directories.

This project is set over a very brief time-frame, needing to be completed by the 11^{th} of May 2018. This means that the software methodology must allow for rapid development to take place, whilst still allowing time for tests and documentation to be produced. Due to this, the most suitable software development methodology will most likely be one which allows for continuous development to take place.

Finally, as this project is being carried out on behalf of another company, it is evident that the development team will not be maintaining the project. Due to this it is critical that the software methodology

allows for testing to take place, as well as providing the ability to produce regular documentation. This will allow the development team that this project gets passed onto to more easily maintain the application we are producing.

2.1.2 Requirements of the methodology from a development team perspective.

From a team perspective it is very important that the sections which the project are broken down to are *non-blocking* and prevent development *deadlocks* from occurring, in which one or more developers are waiting for other developers to complete their work such that they can continue with their own. Ideally the methodology should help prevent these deadlocks, but also be flexible enough to allow for developers to easily swap tasks with one another to easily alleviate stress if such a lock does occur.

2.2 Consideration of Software Development Methodologies

Here we shall consider multiple software development methodologies and how each could be either beneficial or detrimental to the project.

2.2.1 Waterfall Model

The first methodology to be considered is the traditional Waterfall Model [3]. This model involves simplifying the process into five stages: analysis, design, coding, testing and maintenance. Each of these stages are carried out one after another. To successfully use this model the requirements must be in-depth and decided upon completely before the design starts, thus if a change in requirements occurs the waterfall is to be reset.

By having stages dedicated to in-depth requirement design and testing, this methodology would allow for robust testing and accurate documentation which would be useful when handing the project over to the health board. Despite this, as the model has to be reset if a change in requirements occurs, this could be very problematic to the project, as it is likely that client may wish to change some specifications further into development. The model would suppress the ability for the client to provide requests for change.

2.2.2 Incremental Methodology

The Incremental Methodology [3], focuses on breaking the project down into smaller increments, and then carrying out each of these increments as a Waterfall, with each each increment providing more functionality to the project. As each increment is treated as a waterfall, each increment is associated with its own testing and documentation phase. The incremental model makes changing requirements easier, as each increment will have requirements attached to it, as long the a requirement has not been produced yet, it may easily be removed or adapted, similarly new increments can be added in such a manner.

Despite breaking down the project, and simplifying the process of changing requirements, this methodology is very document and test heavy. Given that this is not a high risk project, such as a safety critical system, this would only serve to slow down project development. During each increment the team would need to stop development to produce the required documentation. This could result in one of the earlier

mentioned developmental deadlocks where one developer falls behind, the increment cannot be completed, and thus the next cannot be started.

2.2.3 Spiral Methodology

The Spiral Methodology focuses on combining both design and prototyping, whilst reducing project risk by providing a focus on risk assessment [3]. This methodology is similar to the Incremental model as it breaks the project into smaller segments to allow requirement changes to be simplified, however each segment should have an individual risk assessment. Once a segment is completed, it is treated as a prototype to be improved upon.

This method would be useful as once again the project is broken down, however this time rigorous risk analysis is carried out. In addition to this, as each spiral acts as a prototype, these could be used to easily demonstrate work to the client. The disadvantage with this methodology is that by producing multiple prototypes to display, it is likely that feature creep could occur, resulting in more work being requested than is possible within the time frame.

2.2.4 Extreme Programming

Extreme programming, also referred to as XP [6] is a lightweight methodology which relies on small stories instead of large requirements. The focus of XP is to allow code to be developed at a quicker pace, additionally developers are to write code side-by-side (pair programming), such that they may continuously discuss each others code and edit each others code.

Extreme programming is known for providing very quick results and increasing individual developers knowledge of the project code due to the rigorous peer assessment. Unfortunately, this methodology is restrictive to the development as developers would need to be working in teams to produce code, thus a programmer would not be able to work on any code in their spare time unless another member is present.

2.2.5 Kanban Methodology

Kanban is an agile software development methodology which makes use of applying just-in-time principles against the amount of work in progress to the team's capacity [4]. Kanban makes use of Kanban boards. This allows for the work to be visualised and for the team to plan their workflow.

Tasks are represented as notes on the Kanban board which travel horizontally through *swim lanes*. Each swim lane divides tasks into key areas of the project vertically. Tasks then travel through a series of states. Some common states include "To Do", "In Progress", "Testing", and "Done".

The Kanban model is very flexible, allowing developers to freely pick up tasks and work on them, allowing for continuous delivery to take place and allowing requirements to be adjusted easily by letting the developers simply add another task to the board.

When using Kanban it becomes difficult to produce a specific timeline for the project as tasks should

generally have heavy flexibility around the order in which tasks are completed. With this in mind it becomes difficult to predict when each task will be completed. There is also some potential for heavy feature creep with this methodology.

2.3 Chosen Software Methodology: A hybrid Kanban Incremental Methodology

The team were unable select a singular methodology which we believed was able to meet the needs of this project. Initially the Kanban model was decided to be the most suitable as it allowed for the developers to easily select tasks to work on in a suitable manner, without completely committing a developer to a task. This model makes it difficult to produce a concrete timeline of the project, which is not ideal when the project has specific deadlines which will need to be met.

As a solution to this, the team has decided to produce a hybrid model, making use of characteristics provided by both the Kanban and Incremental models.

This hybrid model would first have the project placed into a few very short increments which allow for a more clear timeline to be produced; however, a single longer increment will focus upon development. The development increment will use a Kanban model within it, rather than a traditional Waterfall model, allowing for the flexibility of the Kanban model to be taken advantage of. When designing this model, it has also been decided that the increments are being used to allow milestones to be more carefully designed, thus unlike in the traditional Incremental model, some increments may begin, and even be completed whilst the development increment is still ongoing.

2.3.1 The Increments to be used

Table 2.1 found below addresses the increments to be used within this project. The team has decided that only five increments will be required for this project. Once again, the goal here is not to produce many specific increments as seen within a traditional Incremental methodology, but rather only a few general increments which allow for a more clear project timeline to be produced.

For information as to when each increment will take place, please refer to the Project Plan chapter of this document.

Table 2.1: Increments to be used within the project.

The Project Increments		
ID	**Increment Title**	**Increment Description**
1	Project Selection and Initial Investigation	This increment focuses on selection of the project, and any initial research required. During this increment an initial meeting will take place with our contact, in which requirements can be discussed. The main technologies to be used within the project should be chosen by the end of this increment.
2	Project Definition and Project Set-up	This increment focuses on deciding upon more concrete project specifications and requirements, alongside this the initial project should be set-up during this increment, such that development can be started.
3	Project Development and Testing	This increment takes place over the majority of the project time span, focusing on developing and testing the application. This increment will make use of Kanban boards over the traditional Waterfall model.
4	Reflection of Project and Production	During this increment, time will be taken for the team to reflect upon the progress of the project and address any major changes which need to be made within the project.
5	Finalisation of Project Testing and Production of Project Documentation	This increment should take place near the end of increment 3, and continue until the end of the project. During this increment a final phase of project testing will be carried out, as well as the production of the completed project documentation.

2.3.2 The Layout of the Kanban Board

Within the Kanban Board we have decided to use seven sections, each task will need to make its way through these sections in order, aiming to be promoted from the first section to the final, one section at a time. The sections to be used in order are as follows; “To do”, “In Progress”, “Documentation”, “Cross-Checking”, “Testing”, “Ready for Integration”, and “Done”. These sections are shown within our example in figure 2.1, and explained within table 2.2. Additionally, in the figure 2.1, swimlanes have been provided to show how these can be used to make the board more readable.

Figure 2.1: An example of how the Kanban Board of Increment 3 will look, made using KanbanTool [21].

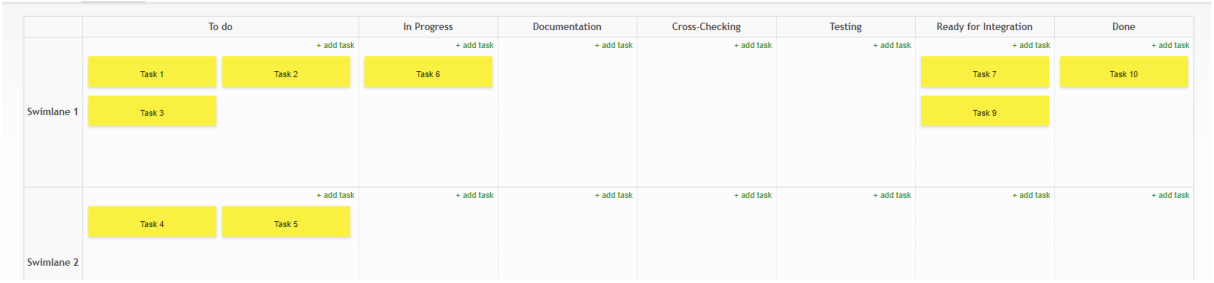

Table 2.2: Sections to be used within the Kanban Board.

Kanban Sections	
Section	**Description**
To do	Tasks placed within this section are yet to be started by a developer, once any work begins on a task it should immediately be promoted from this section to in progress.
In progress	Tasks within this section are currently being developed.
Documentation	Tasks within this section have been developed and are currently undergoing documentation.
Cross-Checking	Tasks within this section are awaiting a code-review, such that the team can agree that the code is suitable. This section helps to ensure that sensible code is being produced. Any issues found are highlighted and passed back to the original developer in the "In Progress" category.
Testing	Tasks within this section have been reviewed, and require testing. If the task fails testing then it is moved back to the "in progress" stage. Once all tests are passed the task should be promoted to ready for integration.
Ready for Integration	Tasks within this section have passed testing and are ready to be integrated into the main project. Typically we shall wait until there are multiple tasks present within this section, and then integrate these as a batch; the sizes and importance's of the tasks will determine how many must be present to being the integration. During this stage, integration and systems testing should be carried out.
Done	Tasks within this section require no further work and should be considered complete.

2.4 Testing Methodology

As this project is being carried out on behalf of another company, it is important that we ensure that the project is tested as thoroughly as possible. If any errors are found after the project is handed over to our client, our team will not be able rectify the issues. Because of this we will implement various forms of testing within this project in an attempt to provide as complete a test coverage as possible.

2.4.1 Unit Testing

Whenever a task is passed into the testing section of the Kanban Board, it should undergo thorough unit testing. These tests should attempt to provide complete code coverage of each individual function implemented for that task. If any pre-existing function tests has been amended it should be re-tested using

a combination of the new test requirements, and also the older requirements where necessary. It should be noted, in general the unit tests should favour black box testing; however, where necessary white box testing should be used to ensure that complete code coverage is achieved. By favouring blackbox testing it will be possible to test client and API tasks separately without needing the two to interact during the unit tests. This helps provide separation between the two projects.

2.4.2 Integration and Systems Testing

When a task is moved into the ready for integration section, and enough tasks are present to justify integrating the tasks, all tasks within the section should undergo both integration and systems testing. The tasks will all be combined into a common version of the project, testing shall then be carried out to ensure that each works correctly within the system, this shall be considered integration testing. Once integration testing is completed for all units, a final test of the system should be carried out ensure that the new functionality works, and any existing functions are still working correctly, this shall act as systems testing.

During systems testing we will also ensure that the project works correctly within a web browser, but also on mobile devices. Mobile devices will be tested using an emulator.

2.4.3 Graphical User Interface Testing

In terms of the Graphical User Interface (GUI) testing, we ourselves will not be able to give confirmation whether the design is visually correct as this choice shall ultimately be made by the client; however, we can ensure that aspects such as navigation, menus, screen size responsiveness, and such are working as intended, therefore such features shall be checked. GUI testing will be carried out during the testing phase of any client tasks which require a change to be made to the interface. For example if a new button is added, we carry out checks to ensure that the button sizes correctly on popular screen sizes, only displays when appropriate, and is triggering the correct on click command. Similarly to systems testing, GUI testing should be carried out not only using a web browser, but also using a mobile emulator.

2.4.4 User Testing

Typically user testing is simply considered as user acceptance testing and is carried out towards the end of a project; however, as we wish to ensure that we provide a project which resembles the clients vision as much as possible, we shall carry this out at multiple stages. During the project we shall attempt to arrange a number of meetings with the client in which we may discuss the current application and allow the client access to the project such that they may use it and provide feedback, allowing us to react to feedback throughout the development increment.

2.5 Development Technologies

Finally, to complete our methodology, we shall now address the technologies which shall be used within this project. As this project consists of two sub-projects, where appropriate we shall address the technologies to be used within the client project and API separately.

2.5.1 Frameworks and Programming Languages

Client

The decision has been made to use the Ionic framework [15]. Ionic is one of the world's most popular cross-platform mobile development technology stacks [15]. Ionic stood out due to its ability to easily transfer between devices smoothly. Additionally, as Ionic focuses on producing progressive web-applications it favours the use of TypeScript and JavaScript for programming, this results in it being easily portable between devices. As this suggests, our client project will be written using TypeScript and JavaScript as these are required for Ionic to be used.

Initially we considered using Xamarin [13] to take advantage of its support within the Visual Studio IDE; however, due to a lack of experience of Xamarin use within the group this was decided against. PhoneGap [1] was also considered due to its growing popularity, and it also supporting JavaScript development; however after a brief investigation we found PhoneGap to be rather restrictive and it was decided against.

API

The API shall be produced using the ASP.NET Core framework [19], with the code being written using C#. This choice was made due to every member of the team having worked on large C# projects, thus all being comfortable with the language. The Core framework was chosen over the standard framework as we are not currently aware what type of server this project will eventually be hosted on, we therefore used Core due to it being cross-platform.

The Node.js [9] framework was considered as it would have allowed for TypeScript and JavaScript to be used on the API, theoretically making development easier as the same languages would be used in each sub-project. Node.js was ultimately rejected as the team felt more comfortable using a C# framework which all members have a wealth of experience with.

2.5.2 Integrated Development Environments

To ensure that the projects do not get polluted with development environment files, the decision has been made to limit the environments in which we may use during development. It should be noted that minor changes may still be made using standard text editors, however major works should be carried out using one of the environments stated.

The favoured IDE will be Visual Studio, this decision has been made due to it's wealth of resources and extensions available for C# applications, as well as the teams experience with the environment. Whilst Ionic is not natively supported by Visual Studio, Visual Studio does contain an optional extension which allows Ionic to be used.

The team has decided that alongside Visual Studio, the editor Visual Studio Code may also be used as long as the TSLint [8] and C# [18] extensions have been installed. This decision was made as the team feels that Visual Studio itself can become difficult to implement on non-windows based systems and the

necessity of the large download reduces portability. Thus by also allowing usage of the lightweight text editor it becomes simpler to access the code on various machines.

2.5.3 Database Selection

For this project, we have decided to use a standard SQL database. When discussing requirements of the project we discovered that the project would make best use of a relational database over a non-relational database. The choice of SQL was made due to team experience with the technology.

2.5.4 Software Testing Tools

Client

In terms of testing for Ionic our team has very limited experience, therefore we have made a decision to use the Jasmine [16] JavaScript testing environment. Despite being a JavaScript based testing environment, this can be used for TypeScript as the TypeScript will be to transpiled to be rendered by the browser; thus, we may use the JavaScript testing environment on the transpiled code. In addition to Jasmine, Apache Cordova [5] will be used to emulate various devices to test the client project.

API

In regards to testing any C# code, the decision was made to use the native TestTools library provided with Visual Studio as this is considered a standard within Visual Studio. This does restrict our ability to carry out the tests on machines without Visual Studio, however this test suite appears to provide the clearest testing mechanisms.

2.5.5 Version Control

The team has decided to use Git to manage the projects version control. This decision was made as Git provides many functions, such as the ability to share a common repository to test within before pushing any code to the master branch; this restriction will be valuable as we can prevent tasks from being committed to the master branch until integration testing is completed. Such abilities resulted in Git being chosen over other version control methods. The Git repository will be managed via GitHub [12] due to preference, however alternatives such as BitBucket [11] were considered.

2.6 Coding and Commenting Conventions

To ensure that code produced is not only understandable by members of the team, but is also understandable by those who may later work on the project, we shall implement both coding and commenting conventions. Unfortunately the client and API use two different programming languages which do not strongly resemble one another, thus two separate sets are required.

2.6.1 Client

The client project shall use the TSLint [8] extension as this will help re-factor the TypeScript code into a readable standard. Unfortunately, the linter does not handle comments, so we have decided that any

complex logic should have a comment attached. Additionally, any methods should have brief comments associated with them to explain their uses, furthermore any classes should provide a description of the purpose of the class.

2.6.2 API

Microsoft provide their own coding and commenting conventions [17] which are recommended to be used with the C# programming language. Our team have opted to follow these as we believe that these conventions are easy to enough to follow. Similarly to the client, the coding standards are not addressed in-depth, so we shall use the same commenting logic here as seen within the client project.

3

Requirements

3.1 Functional Requirements

Table 3.1: Client-Side Requirements

Requirement Id	**Requirement**
CLIENT-REQ1	Each page in the application should have a consistent header for navigation, providing access to the side menu, and to provide additional information about the current page.
CLIENT-REQ2	Each page in the application should have a consistent side menu. This can be used to display additional activities that the user may carry out.
CLIENT-REQ3	A component should be present to display a list of approved exercises that the user can perform in addition to their recommended exercise routine.
CLIENT-REQ4	A way to monitor the users recovery process should be provided.
CLIENT-REQ5	A way to see the users progress at a glance, including statistics.
CLIENT-REQ6	A page to display the users exercises for the day should be present.
CLIENT-REQ7	A visualisation example of the exercises that the user must carry out should be displayed.
CLIENT-REQ8	Methods to log in and log out of the application should be provided.
CLIENT-REQ9	A method should be provided to allow users to register an account.
CLIENT-REQ10	A method should be provided to allow users to request a password reset.
CLIENT-REQ11	When carrying out exercises, users must be able to start and stop the exercises at their will.
CLIENT-REQ12	On timed exercises, the exercise should stop itself if the user has not manually ended the exercise.
CLIENT-REQ13	The user should be able to input progress if necessary (such as reps, sets, time) when completing an exercise.
CLIENT-REQ14	An option should be available to allow users to be advanced further into the recovery process, but only if they have had medical confirmation to do so.
CLIENT-REQ15	The interface should have a main menu page that provides clear, visual navigation of the application.
CLIENT-REQ16	The system should notify a user to remind them to perform exercises.
CLIENT-REQ17	The user should not be able to access the application without an account, except for any information which allows them to log-in or register.

Table 3.2: Server-Side Requirements

Requirement Id	Requirement
SERVER-REQ1	The API should be able to return information regarding exercises at various stages.
SERVER-REQ2	The API should provide functions to store and retrieve information about a users progress.
SERVER-REQ3	The API should be able to store and retrieve information about a users progress (e.g. reps, time taken) for each individual exercise.
SERVER-REQ4	The API should be responsible for managing user authentication.
SERVER-REQ5	The API must be able to retrieve additional approved exercises which the user may carry out at their current stage.
SERVER-REQ6	The API should ensure that it can be easily accessed from the Client project, and provide results in easy to interpret manner.

3.2 Non-Functional Requirements

Table 3.3: Non-Functional Requirements

Requirement Id	Requirement
NF-REQ1	The database should be optimised to reduce storage requirements and data usage.
NF-REQ2	The application should be cross-platform such that it works on Android, iOS, and modern desktop browsers.
NF-REQ3	The application should be protected against common web security vulnerabilities.
NF-REQ4	The application should be a progressive web application using a mobile-first approach.
NF-REQ5	Additional exercises and relevant videos should be able to be added without the need to change source code; however, database changes will be required.
NF-REQ6	The project must be well documented and tested to ease the handover process.
NF-REQ7	Both traditional and user-acceptance testing should be performed.
NF-REQ8	The project should follow the coding and commenting conventions discussed within the methodology chapter of this document.
NF-REQ9	The project will inform users of errors when they occur.

4

Specification

4.1 Project Specifications

Client Interface Specifications

Table 4.1: Client Interface Specifications.

Client Interface Specifications	
Specification Id	**Specification**
CLIENT-SPEC1	This shall be produced using the Ionic framework.
CLIENT-SPEC2	There should be an input field for username and password.
CLIENT-SPEC3	A link should be present that allows users to request a password reset.
CLIENT-SPEC4	A modal should be produced for registering new users.
CLIENT-SPEC5	A common header component will be placed on each page.
CLIENT-SPEC6	The common header should contain the name of the current page.
CLIENT-SPEC7	The common header should contain a button to spawn a side menu containing approved exercises.
CLIENT-SPEC8	The common header should contain a small button that when pressed spawns a modal to allow them to enter a GP code to skip a stage.
CLIENT-SPEC9	The side menu should consist of a list of activities represented via text and icons.
CLIENT-SPEC10	The main menu page will display login details if the user is not logged in, otherwise a navigation menu should be displayed.
CLIENT-SPEC11	The main menu navigation buttons are as follows: "Preform Today's Exercises", "View Statistics", and "Log out".
CLIENT-SPEC12	The side menu and main menu navigation should only be visible if the user is logged in.
CLIENT-SPEC13	When the user presses "Preform Today's Exercises" they shall be directed to the exercise page.
CLIENT-SPEC14	When the user presses "View Statistics" they shall be directed to the statistics section.

CLIENT-SPEC15	When the user presses "Log out", the main menu should be removed and login options should be displayed again.
CLIENT-SPEC16	When on the exercises screen, the user should see today's exercises.
CLIENT-SPEC17	Daily exercises will be accompanied with a video demonstrating the exercise.
CLIENT-SPEC18	A start and stop button should be provided, such that start is pressed to start the exercises and stop is used to complete it early.
CLIENT-SPEC19	The user should be shown how long is left for the current exercise.
CLIENT-SPEC20	If the user stops an exercise early, a modal will be displayed to enter information such as how many reps were completed.
CLIENT-SPEC21	A modal will be displayed when it is required for the user to enter data regarding their exercises once the exercises are completed.
CLIENT-SPEC22	The user should be made aware when they have completed their daily exercises.
CLIENT-SPEC23	A back button should be present such that the user can return to the main menu from the exercise page.
CLIENT-SPEC24	When on the statistical information page, the user will be given information about their exercises and recovery process thus far.
CLIENT-SPEC25	A back button should be present such that the user can return to the main menu from the statistics page.
CLIENT-SPEC26	The client should support the two way binding of data.
CLIENT-SPEC27	The client interface should have an iOS, Android, and web based layout.
CLIENT-SPEC28	The client interface should be responsive, such that it adjusts to the screen resolutions.
CLIENT-SPEC29	The client interface should return an error if it is unable to connect to the API.

Client Logic Specifications

Table 4.2: Client Logic Specifications.

Client Logic Specifications	
Specification Id	**Specification**
CLOGIC-SPEC1	When a successful login has occurred, a cookie should be stored to remember the account.
CLOGIC-SPEC2	If a logout is performed, any related cookies should be cleared.
CLOGIC-SPEC3	A controller should be provided for each page, allowing data to be updated.
CLOGIC-SPEC4	When preparing data for the main menu, data should be retrieved from the API regarding the current user.
CLOGIC-SPEC5	When the user presses the start button on the exercise screen, a timer should begin to dictate how long the user has left.
CLOGIC-SPEC6	When the user presses stop on the exercise screen, or an exercise is completed, the timer is stopped.

CLOGIC-SPEC7	When the exercise is marked as complete the modal should be updated appropriately and the data will need to be passed back to the API.
CLOGIC-SPEC8	The root page must be set to the home page.
CLOGIC-SPEC9	When gathering statistics, the information should be gathered from the API, and calculated on the front end.
CLOGIC-SPEC10	Exceptions should be put into place, such that the system informs the user of an issue if the API cannot be reached.
CLOGIC-SPEC11	Push notifications should be sent to the users device to advise them to carry out exercises.

API Logic Specifications

Table 4.3: API Logic Specifications.

API Logic Specifications	
Specification Id	**Specification**
API-SPEC1	The API should have a unique route for any call which the API can make.
API-SPEC2	The API should contain models for any database interactions.
API-SPEC3	The API should be able to access the database.
API-SPEC4	The API should be able to carry out user authentication.
API-SPEC5	When retrieving from the database, the API should determine the correct phase of the application.
API-SPEC6	The API should spawn an error if connection to the database is lost.
API-SPEC7	The API should return data as JSON strings, such that it can be easily accessed when retrieved by the client.
API-SPEC8	The API must be able to get all additional exercises from the database.
API-SPEC9	The API must be able to return any recommended exercises for a given user from the database.
API-SPEC10	The API must be able to update any tables relevant to a user's exercise when the correct input is provided.
API-SPEC11	It should be possible to generate a Globally Unique Identifier(GUID) that can be used for authorisation to skip exercise phases.

Server and Database Specifications

Table 4.4: Security and Database Specifications.

Security and Database Specifications	
Specification Id	**Specification**
SERVER-SPEC1	The database should be relational and normalised.
SERVER-SPEC2	The database should contain a local link to the exercise videos.
SERVER-SPEC3	The database should contain any information regarding a user's progress.
SERVER-SPEC4	The database should contain information regarding the exercises.
SERVER-SPEC5	The database should contain information of the users account.
SERVER-SPEC6	All data should be sanitised before being used by the database.
SERVER-SPEC7	Usernames entered must be unique and used as a primary key.

Non-functional Specifications

Table 4.5: Non-functional Specifications.

Non-functional Specifications	
Non-functional Id	**Specification**
NONFUNC-SPEC1	Unit testing will be carried out where applicable.
NONFUNC-SPEC2	Documentation will be produced where applicable.
NONFUNC-SPEC3	Integration testing will be carried out on a regular basis.
NONFUNC-SPEC4	Regular meetings with the client should be arranged to allow for user acceptance testing to take place.
NONFUNC-SPEC5	The ASP.NET standard commenting and coding conventions should be followed whilst writing the API.
NONFUNC-SPEC6	The TSLint coding and commenting standards stated earlier within this document should be followed whilst designing the client.
NONFUNC-SPEC7	All tasks should first be implemented within a mobile screen, and then later a standard browser.

4.2 Requirements/Specification Cross-Referencing

To ensure that each requirement is met, each requirement is cross-referenced with one or more specifications to ensure all required functionality will be implemented.

Client-Side Requirements Cross-Reference

Table 4.6: Client-Side Requirements Cross-Reference.

Client-Side Requirements Cross-Reference	
Requirement ID	**Related specification codes**
CLIENT-REQ1	CLIENT-SPEC5, CLIENT-SPEC6, CLIENT-SPEC7, CLIENT-SPEC8
CLIENT-REQ2	CLIENT-SPEC7, CLIENT-SPEC9
CLIENT-REQ3	CLIENT-SPEC9
CLIENT-REQ4	CLIENT-SPEC16, CLIENT-SPEC24, CLOGIC-SPEC4
CLIENT-REQ5	CLIENT-SPEC24, CLOGIC-SPEC9
CLIENT-REQ6	CLIENT-SPEC16
CLIENT-REQ7	CLIENT-SPEC17, CLIENT-SPEC22, CLOGIC-SPEC7
CLIENT-REQ8	CLIENT-SPEC2, CLIENT-SPEC8, CLOGIC-SPEC1, CLOGI-SPEC2
CLIENT-REQ9	CLIENT-SPEC4
CLIENT-REQ10	CLIENT-SPEC3
CLIENT-REQ11	CLIENT-SPEC18, CLIENT-SPEC19, CLOGIC-SPEC5, CLOGIC-SPEC6
CLIENT-REQ12	CLOGIC-SPEC5, CLOGIC-SPEC6
CLIENT-REQ13	CLIENT-SPEC20, CLIENT-SPEC21, CLOGIC-SPEC7
CLIENT-REQ14	CLIENT-SPEC7, API-SPEC11
CLIENT-REQ15	CLIENT-SPEC11, CLIENT-SPEC12, CLIENT-SPEC13, CLIENT-SPEC14, CLIENT-SPEC15, CLIENT-SPEC23, CLIENT-SPEC25, CLOGIC-SPEC8
CLIENT-REQ16	CLOGIC-SPEC11
CLIENT-REQ17	CLIENT-SPEC10

Server-Side Requirements Cross-Reference

Table 4.7: Server-Side Requirements Cross-Reference.

Server-Side Requirements Cross-Reference	
Requirement ID	**Related specification codes**
SERVER-REQ1	API-SPEC2, API-SPEC3, API-SPEC5, API-SPEC9, SERVER-SPEC2, SERVER-SPEC4
SERVER-REQ2	API-SPEC2, API-SPEC3, API-SPEC5, SERVER-SPEC3
SERVER-REQ3	API-SPEC2, API-SPEC3, API-SPEC10, SERVER-SPEC3, SERVER-SPEC4, SERVER-SPEC5

SERVER-REQ4	API-SPEC2, API-SPEC3, API-SPEC4, SERVER-SPEC5, SERVER-SPEC7
SERVER-REQ5	API-SPEC2, API-SPEC3, API-SPEC8, SERVER-SPEC4
SERVER-REQ6	CLOGIC-SPEC3, API-SPEC1, API-SPEC2, API-SPEC3, API-SPEC7

Non-Functional Requirements Cross-Reference

Table 4.8: Non-Functional Requirements Cross-Reference.

Non-Functional Requirements Cross-Reference	
Requirement ID	**Related specification codes**
NF-REQ1	SERVER-SPEC1
NF-REQ2	CLIENT-SPEC1, CLIENT-SPEC28, NONFUNC-SPEC7
NF-REQ3	SERVER-SPEC6
NF-REQ4	CLIENT-SPEC1, CLIENT-SPEC26, CLIENT-SPEC27, NONFUNC-SPEC7
NF-REQ5	CLOGIC-SPEC3, API-SPEC2, SERVER-SPEC1,
NF-REQ6	NONFUNC-SPEC1, NONFUNC-SPEC2, NONFUNC-SPEC3
NF-REQ7	NONFUNC-SPEC1, NONFUNC-SPEC3, NONFUNC-SPEC4
NF-REQ8	NONFUNC-SPEC5, NONFUNC-SPEC6
NF-REQ9	CLIENT-SPEC29, CLOGIC-SPEC10, API-SPEC6

5

Risk Analysis

Risk management is an important aspect of a successful project. To identify and assess factors that may jeopardise the full completion of the project, a risk assessment has been undertaken. Tables 5.2 and 5.3 identify the risks for this project.

An explanation of the scoring numbers used in these tables is presented below in table 5.1. In our risk assessment table a description of the risk is given along with a risk mitigation strategy. The likelihood and potential damage to the project is also given with a score between 1-6. This is then multiplied together to give a risk number.

Table 5.1: An explanation for the risk analysis presented in table 5.2 and 5.3

Risk Analysis Table Explanation				
Risk Chart			**Damage Chart**	
Level 1:	Minor Risk		**Level 1:**	Minor Damage
Level 2:	Low Risk		**Level 2:**	Low Damage
Level 3:	Moderate Risk		**Level 3:**	Moderate Damage
Level 4:	Serious Risk		**Level 4:**	Serious Damage
Level 5:	Massive Risk		**Level 5:**	Massive Damage
Level 6:	Guaranteed Risk		**Level 6:**	Complete Project Wipe
Colour Warning			**Risk Number**	
Green:	Minimum Risk/Damage		**0-11**	
Orange:	Moderate Risk/Damage		**12-23**	
Red:	Massive Risk/Damage		**24-36**	

5.1 Technical Risks

Table 5.2: Technical Risks For The Project

Risk Analysis Table				
ID	**Risk**	**Likelihood**	**Damage**	**Risk Number**
1	**Data Loss/Corruption** A large data loss would result in a huge setback for the project's development, potentially making finishing by the deadline impossible. **Risk Mitigation** Source control will be used to ensure a copy of the project is always available and up to date in a secure, reliable manner.	1	5	5
2	**Software Deployment** Given the teams inexperience with deploying software, especially mobile applications, this may hamper our ability to deploy builds for the stakeholders. **Risk Mitigation** By devoting an appropriate amount of time in the project plan to the deployment process, we can ensure deadlines are met.	2	2	4
3	**Poor Coding Conventions And Best Practices** As we are working as a team of four developers, it is important all members follow similar coding styles and methods. Failure to do this would result in duplicated code, increased bugs, and time wasted merging conflicting changes. **Risk Mitigation** By instructing all team members on a strict set of agreed upon coding conventions to follow, the risk of cluttering the project will be significantly reduced.	3	2	6
4	**Unable To Access Our Host Machine** We are considering hosting our testing server on a university machine which a team member has been given access to. We may run into a scenario where nobody can access the server should that member be unavailable. **Risk Mitigation** By ensuring we research a range of hosting options, and have a contingency plan ready should we need it.	4	4	16
5	**Code Review Bottleneck** To improve the consistency and overall quality of our code, we will be performing code reviews as part of our kanban workflow before code can be merged. As we are only a small team, this could easily create a bottleneck for us if nobody is free to review. **Risk Mitigation** Strong communication between the team is important here to avoid this risk. Having a grace period before major milestones is also important to account for unforeseen slow downs like this.	3	3	9

5.2 Non-Technical Risks

Table 5.3: Non-Technical Risks For The Project

Risk Analysis Table				
ID	**Risk**	**Likelihood**	**Damage**	**Risk Number**
6	**Feature Creep** As we are working with a client during the development of this project, there is a potential that the project may increase in scope beyond what is capable within the time frame. **Risk Mitigation** By adhering to a strict set of requirements and specifications outlined in this document, any risk of feature creep is reduced.	4	3	12
7	**Project Complexity** Given that all members of the team are fairly inexperienced at mobile development, there is a strong possibility of underestimating the complexity of the project. **Risk Mitigation** This risk can be avoided by spending an adequate amount of time researching technologies and similar applications. We also mitigate this risk by carefully outlining a project plan in the final section, giving us a confident breakdown of how long each feature will take to implement.	3	3	9
8	**Short-Term Team Member Illness** In the event of a team member falling ill, such as a cold or fever, the project's development could be held back by up to a week. **Risk Mitigation** The flexibility of the chosen methodology, Kanban, means that workload can be easily adjusted to account for illness or deadlines outside the project. We will also ensure all team members are familiar with both the front-end and back-end projects if important work needs to be picked up.	4	2	8
9	**Lack of QA Resources** As we are only a small development team - without no dedicated tester - it is possible that bugs may slip through that someone else would have picked up on. **Risk Mitigation** We will adopt a test first development style, including UI tests where possible. A clear step in the Kanban workflow will also be dedicated to testing new code before merging.	4	3	12
10	**Being Responsible For NHS/Personal Medical Data** As we are going to be storing some potentially confidential medical data about users, it is important we take security very seriously. **Risk Mitigation** We will be taking security very seriously within our specification and requirements, as well as using only trusted technologies for building the application.	5	5	25

Risk Analysis Table				
ID	**Risk**	**Likelihood**	**Damage**	**Risk Number**
11	**Loss Of Client Contact** Given the busyness of the NHS, and the forced NHS client holidays period, it is possible we will lose client contact for a period of time. **Risk Mitigation** It is important we keep up to date with our client on availability and establish regular contact on both sides about any absence or deadlines.	4	4	16
12	**General Data Protection Regulation(GDPR)** New data protection regulation is being implemented as a part of the data regulation overhaul within the EU. As a result of this, how we handle patient data may need to change depending on how applicable the new regulations are to us. **Risk Mitigation** We will design the system in such a way as to avoid capturing unnecessary patient data, as well as informing the patients of their rights under the new regulations where appropriate.	4	4	16

5.3 Risks in Descending Order

Table 5.4: Technical Risks For The Project

Risks In Descending Order		
ID	**Risk**	**Risk Number**
10	**Being Responsible For NHS/Personal Medical Data**	25
11	**Loss Of Client Contact**	16
12	**General Data Protection Regulation(GDPR)**	16
4	**Unable To Access Our Host Machine**	16
6	**Feature Creep**	12
9	**Lack of QA Resources**	12
7	**Project Complexity**	9
5	**Code Review Bottleneck**	9
8	**Short-Term Team Member Illness**	8
3	**Poor Coding Conventions And Best Practices**	6
1	**Data Loss/Corruption**	5
2	**Software Deployment**	4

6

Project Plan

In this chapter, we present a clear project plan which outlines how time is spent during the project's development. As discussed in chapter 2, the project will follow a hybrid Kanban Incremental methodology. Because of this, the work schedule has been kept flexible within each increment to take advantage of the Kanban methodology. For example, the majority of work done during development is to be developed in parallel during the "Project Development And Testing" increment. This is reflected in the Gantt chart presented later in this chapter.

6.1 Milestones

Milestone	**Deliverables**	**Due Date**
Milestone 1	Methodology and Requirements Document Specification Document	9th February 2018
Milestone 2	Interim Report	23rd March 2018
Milestone 3	Project Demonstration Fair User Manual Design Document Testing Document Narrative and Reflective Account	11th May 2018

6.2 Team Roles

Although all four team members will be working as full-stack developers on the project, we have chosen to select a lead front-end developer, and a lead back-end developer. While these two members will mostly stick to their respective end of the code-base, there is no restriction on which tasks can be picked up on the Kanban board. The major importance of these roles is to easily settle disputes regarding decisions for the code-base. For example, if two team members are in favour of a change on the front-end, but the other two members against, then the lead front-end will have the final say. We have also appointed a member of the team as the main consultant when contacting our client.

We have also made the decision not to appoint a specific team leader. During early decisions on the project we found multiple conflicting opinions on directions to go within the project. We found ourselves

commonly split 50/50 between crucial decisions. This led us to the idea of assigning one group member for each side (front and back), to break these tie-breaker scenarios.

6.3 Final Team Structure

The points made above lead us to the final roles described below:

Liam Miller - Lead Test Architect, Mediator, Full-Stack Developer

As Liam Miller has previously been in employment with the NHS, he has volunteered to be the main mediator when communicating with our client. He has also expressed interest in taking a lead in how we go about testing the application during the project.

James Watson - Lead Front-End Developer, Full-Stack Developer

Given his knowledge in related front-end technologies around the project, James Watson was chosen as the Lead Front-End Developer. While he will mostly stick to implementing the Ionic mobile application, he will still have a large involvement in the full application stack.

Alex Gregory - Lead Back-End Developer, Full-Stack Developer

Because of his strong preference with back-end technologies, Alex Gregory was our natural choice to be our appointed Lead Back-End Developer. He will be working primarily on our C# back-end and making decisions on our server stack.

Joshua Fletcher - Lead Implementation and Deployment Architect, Full-Stack Developer

Joshua Fletcher has shown to have a strong knowledge and interest in the deployment and implementation of our application, including hosting options and securely linking our mobile application to our server. Because of this, he will have a strong involvement in maintaining this area, while also working on the full stack of our application.

6.4 Gantt Chart

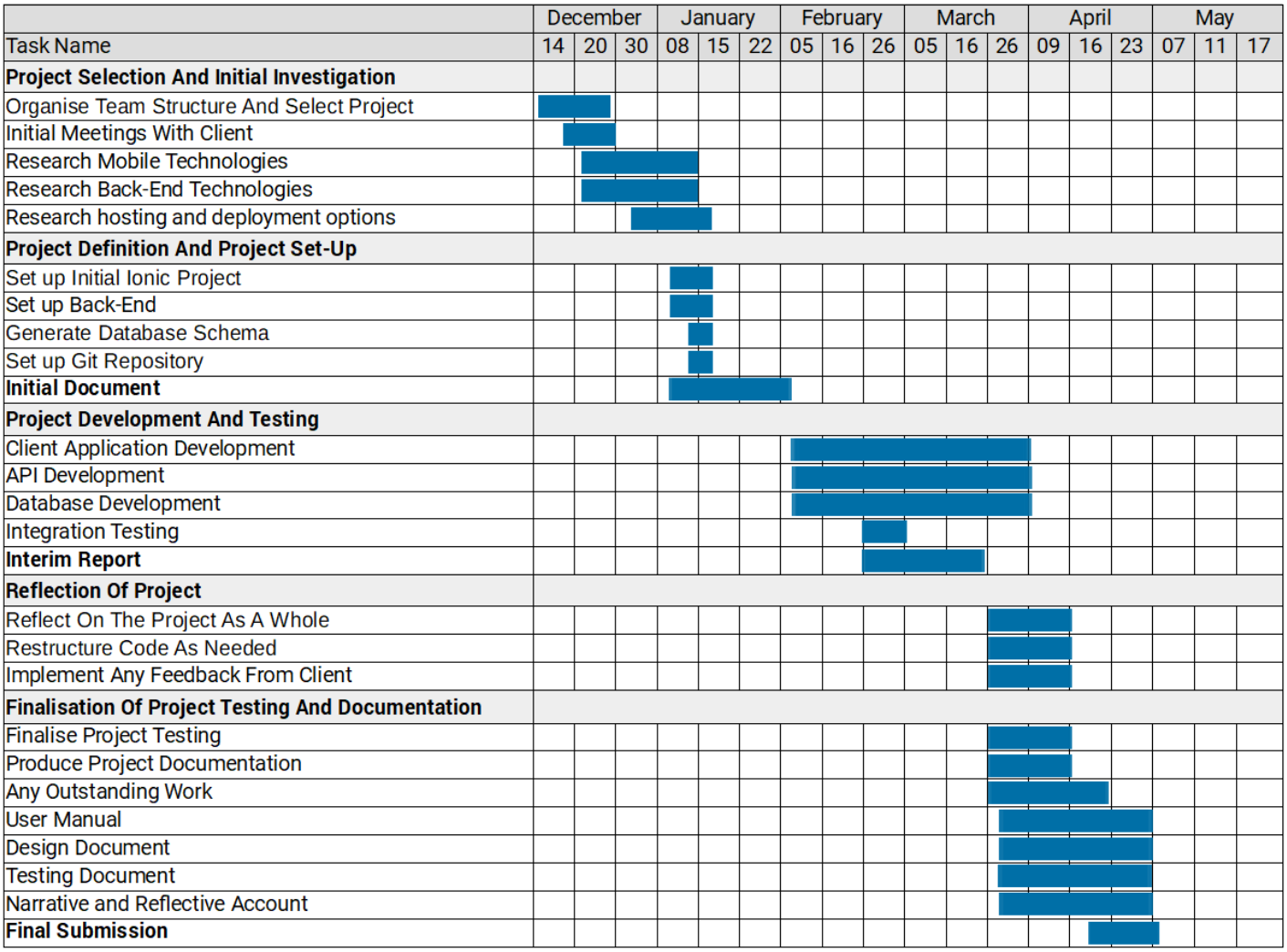

Figure 6.1: Gantt chart showing the work schedule for the project.

As previously mentioned, we have kept the order of which tasks are to be done highly flexible. With the exception of work related to milestones and initial setup, most major work is developed in parallel. This means that we can also take advantage of having four team members, and have each one potentially working on a different feature. This supports the flexible nature of the chosen Kanban methodology.

In order to ensure time is given for reflection, a one week grace period is given before major milestones. As shown in figure 6.1, all tasks are scheduled to be complete one week before their respective deadline. A dedicated period of 3 weeks has also been given to complete all outstanding tasks for the project. As discussed in the risk analysis report in chapter 5, a possible risk involved could be a minor illness of a team member. This one week grace period and outstanding work period doubles up as an excellent safety net for this circumstance.

7

Initial Work

7.1 GitHub Repository

In order for our team to coordinate code releases, we have set up a private repository using GitHub. The configuration of the GitHub project is designed to prevent unsupervised submissions of code as well as direct submissions of code to the master branch, all code must be submitted via a pull review.

A pull review is effectively a code review with the other members of the group looking it over, and ensuring that it will work properly. This will ensure that the code is at least glanced at before submission, and should prevent major errors.

Finally, every member was added to the private repository, so that everyone can remain in line with regards to code. This will avoid issues revolving merging code at later dates.

7.2 Environment Setup

During the course of creating this document, two of our members also configured a cross operating system working environment for developing the application.

The original creation was done on Linux, to catch edge cases with working on Linux. With the additional member working on doing the work on Microsoft after confirming that the project worked on the Linux Environment.

We then later created a development configuration and setup guide, including all of the items and programs that are needed in order for the project to work. This guide included the base programs that are needed for the project, such as NPM and Git. As well as the IDEs that have been considered best suited when working with the project. The aim of this was to make the project quick to set up, both for the original developers of the system, as well as any developers who may need to work on the system in future, as they will need to have a suitable environment for development.

Our cross development work was also done in order ensure that the software could be deployed on both Linux and Windows based systems, this is important as it means the hosting environment is not tied to any one specific operating system.

7.3 Database Setup

As a part of our initial work, we designed a database schema that will meet our needs. As a part of our work, we developed a creation script, an example of which is below.

```
/****** Object:  Table [dbo].[AdditionalExercises]    Script Date: 06/02/2018
    13:40:16 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AdditionalExercises](
  [StageID] [uniqueidentifier] NOT NULL,
  [ExerciseName] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_AdditionalExercises] PRIMARY KEY CLUSTERED
(
  [StageID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

One of the most important aspects of the creation scripts was ensuring that both primary and foreign keys were correctly produced. The creation of a script allows us to all keep local DB copies of the database, meaning that we can all work on the code at the same time without causing any trouble.

It will also allow us to create the database on a third party hosting environment at some point, allowing us to do customer demos in the field. As well as allowing us avoid some of the issues of working with multiple different databases which might change over time.

By working with the script now, it allows us to check the configuration of the database early, and catch any problems that the database may cause our code now, rather than later. This should allow us to avoid building unneeded assumptions into the code.

Bibliography

[1] Adobe. *PhoneGap*. 2018. URL: https://phonegap.com/ (visited on 02/04/2018).

[2] AgeUK. *Older people in Wales: key facts and statistics*. 2009. URL: https://www.ageuk.org.uk/pagefiles/7010/Older_people_in_Wales_key_facts_and_statistics.pdf?dtrk=true (visited on 01/25/2018).

[3] Alshamrani, Adel and Bahattab, Abdullah. "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model". In: *International Journal of Computer Science Issues (IJCSI)* 12 (2015), pp. 106–111.

[4] Altassian. *Kanban - A brief introduction*. 2018. URL: https://www.atlassian.com/agile/kanban (visited on 02/06/2018).

[5] Apache. *Apache Cordova*. 2018. URL: https://cordova.apache.org/ (visited on 02/04/2018).

[6] Copeland, Lee. "Extreme programming". In: *Computerworld* 35.49 (2001). Copyright - Copyright Computerworld Inc. Dec 3, 2001; Last updated - 2017-11-14; CODEN - CMPWAB; SubjectsTermNotLitGenreText - United States; US, pp. 48–46.

[7] Duolingo. *The best new way to learn a language.* 2018. URL: https://www.duolingo.com/ (visited on 01/28/2018).

[8] egamma. *TSLint Extension for Visual Studio Code*. 2018. URL: https://marketplace.visualstudio.com/items?itemName=eg2.tslint (visited on 02/04/2018).

[9] Foundation, Node.js. *Node.js*. 2018. URL: https://nodejs.org/en/ (visited on 02/04/2018).

[10] Government, Welsh. *Health and Social Care Intergration*. 2015. URL: http://gov.wales/topics/health/socialcare/working/?lang=en (visited on 01/25/2018).

[11] Inc, Atlassian. *BitBucket*. 2018. URL: https://bitbucket.org/ (visited on 02/04/2018).

[12] Inc, GitHub. *GitHub*. 2018. URL: https://github.com/ (visited on 02/04/2018).

[13] Inc., Xamarin. *Xamarin*. 2018. URL: https://www.xamarin.com/ (visited on 02/04/2018).

[14] Independent. *Health Service in Crisis*. 2017. URL: http://www.independent.co.uk/news/health/nhs-shortages-funding-straining-at-seams-review-findings-cqc-annual-report-hospitals-mental-health-a7991596.html (visited on 01/25/2018).

[15] Ionic. *Ionic Framework*. 2018. URL: https://ionicframework.com/ (visited on 02/04/2018).

[16] Labs, Pivotal. *Jasmine Behaviour-Driven JavaScript*. 2018. URL: https://jasmine.github.io/ (visited on 02/04/2018).

[17] Microsoft. *C# Coding Conventions*. 2018. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions#commenting-conventions (visited on 02/04/2018).

[18] Microsoft. *C# Extnetnion for Visual Studio Code*. 2018. URL: https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp (visited on 02/04/2018).

[19] Microsoft. *.Net*. 2018. URL: https://www.microsoft.com/net (visited on 02/04/2018).

[20] Online, Wales. *Health Service in Crisis*. 2018. URL: https://www.walesonline.co.uk/news/wales-news/winter-crisis-welsh-nhs-told-14113037 (visited on 01/25/2018).

[21] Shore, Labs. *Kanban Tool - Kanban Boards for Business*. 2018. URL: https://kanbantool.com/ (visited on 02/06/2018).

[22] WelshGovernment. *Waiting Times by Month*. 2018. URL: https://statswales.gov.wales/Catalogue/Health-and-Social-Care/NHS-Hospital-Waiting-Times/Diagnostic-and-Therapy-Services/waitingtimes-by-month (visited on 02/05/2018).

List of Figures

List of Tables