University of Plymouth

School of Engineering, Computing, and Mathematics

COMP3000

Final Stage Computing Project

2020/2021

ICU Tracker App

Samuel Jordan

10611160

BSc (Hons) Computer Science

# Acknowledgements

# Abstract

Contents

[Acknowledgements 2](#_Toc69730892)

[Abstract 2](#_Toc69730893)

[Statement of Word Count 4](#_Toc69730894)

[Code Submission 4](#_Toc69730895)

[1 Introduction 4](#_Toc69730896)

[1.1 Background 4](#_Toc69730897)

[1.2 Objectives 4](#_Toc69730898)

[1.3 Deliverables 4](#_Toc69730899)

[1.4 Report Overview 4](#_Toc69730900)

[2 Method of Approach 4](#_Toc69730901)

[3 Legal, Social and Ethical 4](#_Toc69730902)

[3.1 Social Issues 4](#_Toc69730903)

[3.2 Legal Issues 4](#_Toc69730904)

[3.3 Ethical Issues 4](#_Toc69730905)

[4 Project Management 5](#_Toc69730906)

[4.1 Methods of Project Management 5](#_Toc69730907)

[4.2 Project Management technologies 5](#_Toc69730908)

[4.3 GitHub 5](#_Toc69730909)

[4.4 Trello 5](#_Toc69730910)

[4.5 Sprints 5](#_Toc69730911)

[4.6 DevOps/Development Pipeline 5](#_Toc69730912)

[5 Requirements 5](#_Toc69730913)

[5.1 Functional Requirements 5](#_Toc69730914)

[5.2 User Requirements 5](#_Toc69730915)

[5.3 Time Management 5](#_Toc69730916)

[6 Research 5](#_Toc69730917)

[6.1 Fitness App 5](#_Toc69730918)

[6.2 Technologies 5](#_Toc69730919)

[6.3 Accessibility 6](#_Toc69730920)

[7 System Architecture and Design 6](#_Toc69730921)

[7.1 Planning documents 6](#_Toc69730922)

[7.2 UML 6](#_Toc69730923)

[7.3 ERD 6](#_Toc69730924)

[7.4 PID 6](#_Toc69730925)

[7.5 User Stories 6](#_Toc69730926)

[7.6 Risk Assessment 6](#_Toc69730927)

[7.7 Interactive Storyboard 6](#_Toc69730928)

[7.8 Technologies 6](#_Toc69730929)

[7.8.1 SQL Server 7](#_Toc69730930)

[7.8.2 ASP.NET Web API 7](#_Toc69730931)

[7.8.3 Insomnia 8](#_Toc69730932)

[7.8.4 Xamarin 8](#_Toc69730933)

[7.8.5 IIS 8](#_Toc69730934)

[7.8.6 Third-Party Technologies 8](#_Toc69730935)

[8 Development 9](#_Toc69730936)

[8.1 Sprint Planning 9](#_Toc69730937)

[8.2 Sprint 1 9](#_Toc69730938)

[8.3 Sprint 2 9](#_Toc69730939)

[8.4 Sprint 3 9](#_Toc69730940)

[8.5 Sprint 4 9](#_Toc69730941)

[8.6 Sprint 5 10](#_Toc69730942)

[8.7 Sprint 6 10](#_Toc69730943)

[8.8 Sprint 7 10](#_Toc69730944)

[9 Testing 10](#_Toc69730945)

[9.1 User Testing 10](#_Toc69730946)

[9.2 UX Testing 10](#_Toc69730947)

[9.3 Insomnia 10](#_Toc69730948)

[10 Project Post-Mortem 10](#_Toc69730949)

[10.1 What Went Well 10](#_Toc69730950)

[10.2 What Didn’t Go Well 10](#_Toc69730951)

[10.3 What Would Be Changed? 10](#_Toc69730952)

[11 Conclusion 10](#_Toc69730953)

[12 References 10](#_Toc69730954)

[13 Appendices 10](#_Toc69730955)

# Statement of Word Count

# Code Submission

# 1 Introduction

## 1.1 Background

## 1.2 Objectives

## 1.3 Deliverables

## 1.4 Report Overview

# 2 Method of Approach

Talk about the use of sprints and agile

# 3 Legal, Social and Ethical

## 3.1 Social Issues

## 3.2 Legal Issues

Data storage

One big legal issue to keep in mind is the storage of data for the app, as the users name and location is being stored which could be an issue if there were to be security problems.

As this is not a fully-fledged app, security is not a huge priority, however some basic security measures were put place to keep it relatively secure.

As the app will be downloaded only for those in ICU, there is also less of a problem for unauthorized access.

## 3.3 Ethical Issues

UX ethical issues due to ICU

Care must be taken to ensure that the app is usable for people with a multitude of different physical and/or mental handicaps, especially as the app is being used in the ICU of a hospital. Failure to make it accessible could make it extremely hard to use, or even completely unusable for some users.

# 4 Project Management

## 4.1 Methods of Project Management

## 4.2 Project Management technologies

## 4.3 GitHub

## 4.4 Trello

Trello was used to be able to keep plan and keep track of everything that I needed to do regarding the app.

## 4.5 Sprints

Sprints are used within the project to make it more manageable and consistent.

The sprints created for the project are in 2-week blocks,

Trello was used to detail the sprints in each of their fortnightly blocks.

Dates were also added to each item in the sprints to be able to split the work down further.

## 4.6 DevOps/Development Pipeline

# 5 Requirements

## 5.1 Functional Requirements

## 5.2 User Requirements

## 5.3 Time Management

# 6 Research

As I had not previously created a mobile app, a lot of research needed to be done to look at what is expected from a modern-day mobile app.

## 6.1 Fitness App

Research was done into both the features and overall aesthetic of current day fitness apps, as the app I was creating was very similar.

Two major leading apps were looked at, Nike Run Club and Google Fit, with Google Fit being more similar to the ICU app being created. A document was first created, including screenshots of all of the important pages to the app. I then went through both apps, noting down their pros and cons, looking at what could be a good addition to my app or a feature that might instead hinder it (Table 1, Table 2).

|  |  |
| --- | --- |
| Pros | Cons |
| Simple to navigate | A lot of empty space |
| Features are sorted on left menu bar | Unselected text is hard to see |
| Challenge’s feature – can add dates |  |
| Simple contrasting colours |  |
| Font is big and easy to see |  |

Table 1. Nike Run Club pros and cons.

|  |  |
| --- | --- |
| Pros | Cons |
| Selection of dark/light mode | Dropdowns are hinted in wrong places |
| Home screen shows quick information | Some text can be hard to read |
| Drop downs used when possible | Text size is a little small (could be due to device settings) |
| Colours are used to accent clickable items |  |
| Buttons are labelled |  |
| Intuitive placement for items |  |
| Settings allows customization |  |
| Home chart page has a radial chart and text – either can be used |  |

Table 2. Google Fit pros and cons.

These evaluations of current apps made a big impact to the features I decided to add to the final app, most notably the use of colours and font sizes.

## 6.2 Technologies

Research was done on the

## 6.3 Accessibility

# 7 System Architecture and Design

## 7.1 Planning documents

## 7.2 UML

A basic UML was created of the API to assist in the development of it.

The UML contains all of the models and controllers as well as the rough structure of the API.

The UML is seen in Figure X

## 7.3 ERD

An ERD (Entity Relation Diagram) was created so that creation of the database could be straightforward, with all of tables and relationships being designed before implementation.

Had to think about all of the data fields that the app would need to use and have access to.

Had to think about what relationships needed to be created so that the app wouldn’t have data duplicates.

Needed to think about redundancy in data, especially with data laws not allowing non-functional data to be stored.

## 7.4 PID

## 7.5 User Stories

## 7.6 Risk Assessment

A risk assessment was made to identify the risks that might be involved in the project, if any of them are likely to happen and if so, how I could mitigate the impact of these.

Table X shows all of the identified risks, their likelihood of happening and what can be done to reduce the probably of them occurring.

## 7.7 Interactive Storyboard

Figma was used to be able create a storyboard for my app.

Figma allows for the creation of “Interactive Storyboards” which means that the user can use the buttons as if the app were created. This meant that I could get some front-end feedback early on in the design stage without having to create the app and then get the feedback which could potentially take months.

## 7.8 Technologies

A number of technologies were used within my project to create the tracker app.

All of the technologies used were researched previously, to make sure that they would work with the app.

A number of these I had used before, however a lot of them I had no prior experience with, so a lot of time was spent researching and performing exercises to learn how they work.

A lot of the technologies chosen were due to compatibility and ease of use. Xamarin, SQL Server and ASP.Net Web API are created by Microsoft, so have a lot of documentation on their use together.

### 7.8.1 SQL Server

SQL server was used as a database to store all the data for the app including the exercises, images and user information. This database was created upon the previously made ERD which specified all the tables and formatting for the entries, which made creating the database relatively easy.

SQL Server was used mainly due to its compatibility with Xamarin and the ASP.Net Web API with all three being developed and supported by Microsoft. This meant that there was a lot of online documentation regarding using these technologies together, which made it a lot easier to develop for. SQL Server is also very widely supported which meant any problems that occurred while developing with it were easy to fix, as there are a multitude of online resources for help. The amount of documentation also means that moving the database in the future, which would be necessary for it to be fully functional, should be a lot easier than with other database technologies.

I also have prior experience using SQL Server, which made designing and creating the database a lot easier as I had developed with it several times before. I also had knowledge of its pros and cons (Table 3) and knew that SQL Server would be perfect for my app.

|  |  |
| --- | --- |
| Pros | Cons |
| Easy to create queries | Debugging can be tricky |
| Good range of data types | Hard to setup first time |
| Support for data migration | Can be costly |
| SQL language easy to understand | Doesn’t support some newer technologies |
| Widely supported and used |  |
| Has a lot of documentation and online materials |  |
| Scales well |  |
| Easy to maintain |  |

Table 3. Pros and cons to the use of SQL Server

### 7.8.2 ASP.NET Web API

The ASP.NET Web API is a framework built upon .NET which allows for the building of HTTP based services such as the API needed for this app.

The API consists of a context file, model files and controller files.

The context file is used to specify relationships between the models.

The model files are used to specify what the database tables are made up of so that the data is in the correct format.

The controllers are used to specify what data can or cannot be requested from the API and how the API gets that data from the database.

Using this meant that I was able to create the database and then from that, the API could access the database, see what data needed to be accessed and then created the models itself. From here the controllers would just need to be made, allowing accessing of the data.

### 7.8.3 Insomnia

Insomnia is an API design platform that allows for quick and easy testing of API functions.

Insomnia was used a lot within my project to test API functions to ensure that they were returning the right information. It was also used to check that it accepted the correct data, stored it appropriately and return the correct codes.

Image X shows all of the calls that I used added in insomnia to test the API.

### 7.8.4 Xamarin

Xamarin is an open-source platform created by Microsoft based upon the .NET framework which uses XAML and C# to allow for cross-platform apps to be created for Android, IOS, MacOS and Windows from one central codebase.

Xamarin also handles the emulation of an android device natively on Windows alongside the Android SDK. This meant that the creation of pages and features was quick as it allowed me to create and test the app locally, without having to export the files to an android device. Xamarin also has a XAML hot reload feature which meant that I was able to quickly change something on the front-end code and it would be updated in the android emulator without having to reload the OS or rebuild the app.

Xamarin was used inside of Visual Studio to code the app, which allowed for the automatic download of extensions and updates.

Xamarin has a number of built in extensions such as Xamarin Forms which was used to create the front end of the app.

Xamarin allows for the changing of the files within each of the native OS’s, in this case android and IOS, as well as the Xamarin project as a whole. This meant that the app could be mostly developed in Xamarin, but for device specific functions such as where images are stored, these could be placed manually within the app files of the OS.

### 7.8.5 IIS

IIS is a function build directly into windows that allows me to host my previously made ASP.Net Web API locally,

If the app were to be deployed publicly and used within the ICU, then the API could easily be moved onto a cloud platform such as Azure to allow it to be accessed from anywhere.

IIS was used alongside the ASP.Net Web API to locally store and access the API.

### 7.8.6 Third-Party Technologies

A number of third-party extensions were used within Xamarin to allow for extra functionality (Table X)

* Microcharts
* Loading
* Flurl

Microcharts was used to be able to display a chart inside of Xamarin as it has to native library to handle diagrams or charts. Microcharts has a good amount of documentation and examples and looks nice, fitting the aesthetic of the app.

Flurl was used to connect the app to the API, handling the HTML requests to the API as well as the storing of the data once it had been received.

Flurl was fundamental in the building of the app as without it, the API calling process would have taken a lot longer to develop and would have been much more complex.

Loading was used as Xamarin has no way to display to the user that an item is loading or unavailable. Loading allows me to block the screen so that the user isn’t able to interact with it while some data is loading as this could cause it to stop working, the user to change data that they didn’t mean to or cause the app to crash. Loading also has a visual element which notifies the user that the data or something on the page is loading which is extremely useful.

# 8 Development

For development of the app, sprints were used to split the load and make it more manageable. These lasted for 2 weeks each and consisted of tasks to do for that fortnight, with each task having their own individual due date and description.

The use of sprints helped to split the load and keep development of the app on track throughout the year.

## 8.1 Sprint Planning

Created the sprints to make sure that everything would be completed by certain dates etc.

## 8.2 Sprint 1

Front-end development and front-end design feedback

## 8.3 Sprint 2

To begin creating the database, SQL Server was first installed onto my computer. This included SQL Server Management Studio (SMSS) which comes preinstalled with SQL Server and is officially supported which meant I could use to create and run queries for the database.

When creating the database, I stored all the queries inside of an SQL file, which would make it easier to recreate the database if necessary. It also meant that I could see all of the columns with their data types easily from one single file. This same SQL file was also used to store all the queries for entering test data into the database. With all of these queries I could recreate the database, including the data by running a single file which was extremely useful. This SQL file was used multiple times throughout development to reset the data for testing the app.

Following creation of the database, the Web API could be created, using the database as a template for the API’s models.

Web API creation

Connecting database to the app

## 8.4 Sprint 3

Finishing of main application

Research into disabilities etc

## 8.5 Sprint 4

Final app changes

User testing of the whole application

## 8.6 Sprint 5

Final app changes and further changes.

## 8.7 Sprint 6

Web accessibility testing

Bug fixes

Final app changes.

## 8.8 Sprint 7

Application fact sheet

# 9 Testing

## 9.1 User Testing

User testing was completed a number of times throughout development

## 9.2 UX Testing

UX testing was completed both on the interactive storyboard as well as once the front-end of the app had been fully developed.

## 9.3 Insomnia

As described in 7.8.3, Insomnia isa program I used to be able to test the functionality of my API.

# 10 Project Post-Mortem

## 10.1 What Went Well

All of the planning went perfectly

Development of the front end was good

## 10.2 What Didn’t Go Well

Development of the API and backend in general was a struggle as I hadn’t used Xamarin or ASP.Net Web API before so it took a lot longer to develop then it should have.

Once I got my head around it, it was fine, but it was a big delay to the development of the app that could have been spent on implementing optional features.

Testing of the app was difficult to do due to COVID, meant that in person testing was basically not possible so an alternative method had to be found. This was made harder by the fact that the API and database were stored locally, meaning users testing the app would have to install a VPN to connect to my PC as well as the app itself.

## 10.3 What Would Be Changed?

A lot of the extra functionalities noted in the user stories would have been good additions to the app.

Extra settings to help with the accessibility of the app would be a good addition. Allowing the user to change the colours, or the font size could be extremely helpful for some users, especially in the ICU.

However, with the limited previous experience with the technologies these were hard to implement, and further knowledge of the technologies would be needed to implement a number of them. If I were to create the app again, the whole process would be a lot quicker as I now have a lot more knowledge of the technologies used.

# 11 Conclusion

# 12 References

# 13 Appendices